Exploiting non-canonicity in the sequent calculus

Vivek Nigam

École Polytechnique
Thèse de Doctorat
Spécialité Informatique

Exploiting non-canonicity in the sequent calculus

Présentée et soutenue publiquement par

Vivek Nigam

le 18 septembre 2009

devant le jury composé de

Rapporteurs:	Jean-Marc Andreoli
	Roy Dyckhoff
Directeur de thèse:	Dale Miller
Examinateurs:	Iliano Cervesato
	Ian Mackie
	Damiano Mazza
	Elaine Pimentel

This work was supported by the Information Society Technologies programme of the European Commission, Future and Emerging Technologies under the IST-2005-015905 MOBIUS project.

Acknowledgments

First of all, I thank my supervisor Dale Miller for his continuous support in the past three years not only with good advice, but also with the encouragement to proceed with my research. He was always available for discussions, correcting me whenever I proposed half-baked ideas and sharing patiently his insights and vision. Most of the results in this thesis have in some way or another been also influenced by Dale and his work.

My gratitude also goes to Jean-Marc Andreoli and Roy Dyckhoff for taking some of their precious time to review this document. I am also grateful to Elaine Pimentel, Ian Mackie, Iliano Cervesato and Damiano Mazza for accepting to be part of my jury.

Special thanks to my colleagues at LIX, in particular, to the members of the Tarski and Church office, David Baelde, Olivier Delande and Alexis Saurin, as well as to the members and collaborators of the PARSIFAL team, Lutz Strassburger, Kaustuv Chaudhuri, Elaine Pimentel, Alwen Tiu, Stefan Hetzl, Agata Ciabattoni, Nicolas Guenot, Ivan Gazeau, Alexandre Viel, Anne-Laure Poupon, Stéphane Lengrand and François Wirion for their passion for logic and proof theory and to members of the other research teams, Ulrich Herberg, Romain Beauxis, Sylvain Pradalier, Miki Hermann, David Savourey, Mário Sérgio Alvim, Juan Antonio Cordeiro, Lisa Allali, Simon Kramer and Frank Valencia for the pleasant time we spent together. I am also grateful to Carlos Olarte for co-organizing the several editions of the *fête sud-americaine du LIX*. Without École Polytechnique's administrative staff, I would have certainly not lasted long in France. Therefore, I must thank Isabelle Biercewicz, Catherine Moreau and Audrey Lemarechal for helping me through the complicated French paper work. I also thank the 296 stair steps of École Polytechnique, which, despite of the wonderful French delicatessen, kept me in good shape.

Finally, I thank from the bottom of my heart my parents and my brother for always supporting me and, in special, my Andrea for bringing joy and color to my life.

Abstract

Although logic and proof theory have been successfully used as a framework for the specification of computation systems, there is still an important gap between the systems that logic can capture and the systems used in practice. This thesis attempts to reduce this gap by exploiting, in the context of the computation-as-proof-search paradigm, two non-canonical aspects of sequent calculus, namely the *polarity assignment in focused proofs* and the *linear logic exponentials*. We exploit these aspects in three different domains of computer science: tabled deduction, logical frameworks, and algorithmic specifications.

This thesis provides a proof theoretic explanation for tabled deduction by exploiting the fact that in intuitionistic logic atoms can be assigned arbitrary polarity. A table is a partially ordered set of formulas and is incorporated into a proof via multicut derivations. Here, we consider two cases: the first case is when tables contain only *finite successes* and the second case is when tables may also contain *finite failures*. We propose a focused proof system for each one of these cases and show that, in some subsets of logic, the only proofs and open derivations available in these systems are those that do not attempt to reprove tabled formulas. We illustrate these results with some examples, such as simulation, winning strategies, and *let polymorphism* typechecking.

We show that linear logic can be used as a general framework for encoding proof systems for minimal, intuitionistic, and classical logics. First, we demonstrate that with a *single* linear logic theory, one can faithfully account for natural deduction (normal and non-normal), sequent calculus (with and without cut), natural deduction with general elimination rules, free deduction and tableaux proof systems by using logical equivalences and different polarity assignments to meta-level literals. Then we exploit the fact that linear logic exponentials are not canonical and propose linear logic theories that faithfully encode different proof systems; for example, a multi-conclusion system for intuitionistic logic and several focusing proof systems.

For the last contribution of this thesis, we investigate what type of algorithms can be expressed by using linear logic's non-canonical exponentials. In particular, we use different exponentials to "locate" multisets of data, and then we show that focused proof search can be precisely linked to a simple algorithmic specification language that contains while-loops, conditionals, and insertion into and deletion from multisets. Finally, we illustrate this result with several graph algorithms, such as *Dijkstra's algorithm* for finding the shortest distances in a positively weighted graph and an algorithm for checking if a graph is bipartite.

Contents

Chapter 1

Introduction

It is unfortunate that computer artifacts are often constructed in such an *ad hoc* fashion. Software is usually just built to run efficiently without too many errors. Other important concerns, that are essential for the quality of large, complicated, and expensive systems, are left aside or when addressed, they are not handled with formal and precise methods: for example, *correctness* – if programs really compute what was intended; *modularity* – if different programs can be composed without affecting their individual behaviors; *readability* – if one can easily understand the logic used in a program to solve a problem; *interoperability* – if programs can be used on different machine architectures; etc. As logic and proof theory have a long tradition of dealing with formal languages and provide powerful tools for reasoning over logic specifications, computer scientists have successfully used logic to specify computation systems. For instance, in the past decades, proof theory has been widely used in the design of programming languages along two lines of research related to the *Curry-Howard isomorphism* [Howard 1980] and the *computation-as-proof-search* paradigm.

Although in the past years many techniques and fundamental results about logic have been discovered, allowing one to specify a wider range of computational behaviors, there is still an important gap between the systems that logic can formally capture and the systems used in practice. For example, in the context of computation-as-proof-search, logic interpreters, such as Prolog, often contain non-logical elements, like *tabling* mechanisms [Ramakrishna 1997, Pientka 2005], that are necessary for expressing more algorithms or for increasing *proof search efficiency*. In order to fill this gap, one needs to further investigate and better understand logic and proof theory. In particular, we are interested in some of their non-canonical aspects that provide us with an opportunity to specify more computations. In this thesis, we exploit two such non-canonical aspects, namely the *polarity assignment in focused proofs* and the *linear logic exponentials*, by using the computation-as-proof-search paradigm in different domains of computer science.

At the heart of the connection between proof search and computation lies the *sequent calculus*, proposed by Gentzen in 1935 [Gentzen 1969]. A sequent calculus system is a collection of *inference rules* that derives multisets of formulas called *sequents*[1]. The computation-as-proof-search paradigm connects computation with the search for sequent calculus proofs as follows: logic formulas represent program instructions; sequents represent states of the *world*; and sequent calculus proofs represent computation traces. The sequent at the root of a sequent calculus derivation represents the initial state of the world and the sequents at its open leaves represent the states of the world that result from performing a sequence of computation steps. Therefore, from this perspective, one is usually interested in specifying a computation, say an algorithm, in such a way that the proofs obtained from its logic specification correspond to its execution runs and vice versa. However, sequent calculus systems without any particular discipline on how inference rules are applied allow too many proofs to be useful for specifying interesting behaviors. One needs to consider some type of *canonical proofs* that have some proof search discipline. Andreoli [Andreoli 1992] proposed such a discipline, called *focusing*, where proofs are structured in two alternating focusing phases and formulas are classified as *positive* and *negative* according to which phase the introduction rules for their main connectives belong to. However, as the atomic formulas do not have inference rules, they are classified arbitrarily as positive or negative. This non-canonical aspect in focusing can be exploited by computer

[1] In fact, Gentzen originally defined sequents as a sequence of formulas.

scientists. For example, changing the polarity assignment for atoms allows one to mix *forward and backward chaining style proof search* and this can be used to capture different computational behaviors [Liang 2008, Liang 2007, Miller 2007a, Jagadeesan 2005, Nigam 2008b].

Linear logic is a substructural logic proposed by Girard [Girard 1987], which has been widely used in both proof theory, as the *logic behind logics*, and computer science, as the *logic of resources*. Differently from classical logics, the structural rules for weakening and contraction are not allowed for all formulas, but only those formulas marked with the so called *exponentials*. However, these exponentials are not canonical in the sense that one could create different colored exponential-like connectives, say *red* exponentials and *blue* exponentials, that are not *equivalent*. In fact, it is possible to have as many exponential-like operators, called *subexponentials*, as one would like: they may or may not allow weakening and/or contraction and they can be organized into a preorder that specifies which operator logically entails others [Danos 1993].

In this thesis, we exploit the non-canonical aspects discussed above in the three following domains of computer science:

Tabled Deduction – Consider attempting to prove the conjunctive query $B \wedge C$ from a logic program Γ. This attempt can be reduced to first attempting to prove B from Γ and then C from Γ. It might well be the case that during the attempt to prove C, many subgoals need to be proved that were previously established during the attempt to prove B. Of course, if proved subgoals can be remembered from the first conjunct to the second, then it might be possible to build smaller proofs and these might be easier to find and to check for correctness. Some implemented logic programming systems introduce *tables* as a device to store proved literals so that their provability can be used in later attempts to prove goals. Systems such as XSB [Ramakrishna 1997] and Twelf [Pientka 2005] make it possible to specify that some predicates should be *tabled*: that is, whenever an atomic formula with such a predicate is successfully proved, that atomic formula is remembered by placing it in a global table. In this way, if the prover attempts to reprove an atom that is already tabled, then the proof process can be immediately stopped with a success. Besides this aspect of not proving formulas, tabling systems improve proof-search considerably by also remembering *finite failures*. Whenever a goal is shown not be provable, it is also stored in a table, and if the interpreter attempts to prove a tabled finite failure, it does not proceed.

Although it is clear that one can build implementations with tabling mechanisms, a more interesting question is whether we can avoid reproving formulas and avoid proceeding when a finite failure is encountered by *proof theoretic means*, that is, enforce in the logic that all available proofs and open derivations are those that do not attempt to prove tabled formulas. We exploit the fact that polarities of atoms in intuitionistic logic can be assigned as positive and negative to specify *focusing disciplines* that accomplish this goal. First we consider the case when a table contains only atomic formulas denoting finite successes. Then, we consider a second case where we also allow tables to contain *fixed point literals* and *universally quantified fixed point literals* denoting finite successes and finite failures.

Logical Frameworks – Logics and type systems have been exploited in recent years as frameworks for the specification of deduction in a number of logics. The most common such *meta-logics* and *logical frameworks* have been based on intuitionistic logic (see, for example, [Felty 1988, Paulson 1989]) or dependent types (see [Harper 1993, Pfenning 1989]). Such intuitionistic logics can be used to directly encode natural deduction style proof systems.

In a series of papers [Miller 1996, Pimentel 2001, Miller 2002, Miller 2004, Pimentel 2005], Miller & Pimentel used classical linear logic as a meta-logic to specify and reason about a variety of sequent calculus proof systems. Since the encodings of such logical systems are natural and direct, the meta-theory of linear logic can be used to draw conclusions about the object-level proof systems. For example, in [Miller 2002], a decision procedure was presented for determining if one encoded proof

system is derivable from another. In the same paper, necessary conditions were presented (together with a decision procedure) for assuring that an encoded proof system satisfies cut-elimination. This last result used linear logic's dualities to formalize the fact that if the left and right introduction rules are suitable duals of each other then non-atomic cuts can be eliminated.

In this thesis, we continue this line of research in two different ways. First, we exploit the fact that the linear logic literals can be assigned arbitrary polarities, and we show that, by using a *single* linear logic theory, one can *faithfully* account for natural deduction (normal and non-normal), sequent calculus (with and without cut) [Gentzen 1969], natural deduction with general elimination rules [von Plato 2001], free deduction [Parigot 1992] and tableaux proof systems [D'Agostino 1994, Smullyan 1968a]. In the second direction, we exploit the fact that the linear logic exponentials are not canonical and propose theories that encode different proof systems, for example a *multi-conclusion system* for intuitionistic logic [Maehara 1954] and several focusing proof systems [Herbelin 1995, Dyckhoff 2006, Liang 2008].

Algorithmic Specifications – A major obstacle to describing algorithms using linear logic programs, in the sense that the set of proofs and of computations runs are in one-to-one correspondence, is that data encoded into contexts does not support enough tests on data. While it is possible in linear logic to detect that the whole multiset of *linear formulas*, that is, formulas that cannot contract or weaken, is empty, it is not possible to perform this test on some particular subset. Consider, for example, that we encode in linear logic a graph with nodes N and adjacency relation A with the following multisets of linear logic atoms:

$$\{node\ x \mid x \in N\} \cup \{adj\ x\ y \mid \langle x, y\rangle \in A\},$$

where *node* and *adj* are predicates. Linear logic provides a simple mechanism to detect that both the set of nodes and the adjacency information are empty, but the logic does not provide means to check emptiness of just N or just A.

We exploit the fact that the linear logic exponentials are not canonical to "locate" data by using *subexponentials*. These subexponentials provide linear logic specifications with enough checks on data to allow for a range of algorithms to be emulated *exactly* via (focused) proof search. We illustrate this claim by specifying a simple programming language, called Bag, containing loop instructions, conditionals and operations that insert into and delete from a multiset, which is powerful enough to specify complicated algorithms, such as *Dijkstra's algorithm* for finding the shortest distances in a positively weighted graph. We then show that for any Bag program there is a one-to-one correspondence between the set of its (partial) computations and the set of (open) focused derivations of its logic interpretation.

1.1 Outline

This thesis is structured as follows:

Chapter 2 introduces the basic concepts and vocabulary used throughout this thesis. It introduces sequent calculus proof systems for classical, intuitionistic and linear logics, highlighting some non-canonical aspects present in these formalisms. In this chapter, we also discuss focusing which is one of the cornerstones of this thesis.

Chapter 3 contains the results related to the declarative specification for tabled deduction. We start this chapter by introducing the focused proof system for intuitionistic logic *LJF* [Liang 2008, Liang 2007], which will be used to derive specialized proof systems that adopt the different focusing disciplines mentioned above. As cuts play an important role in incorporating tables into proofs, we discuss the design of focused proofs with cuts. We then propose the first focused proof system

enforcing that the only proofs available are those that do not reprove *atoms* that are in a table. Later, we introduce a proof theory for fixed points and propose the second focused proof system enforcing that *fixed point literals* and *universally quantified fixed points* in a table are not reproved and that derivations attempting to prove a negative fixed point are not allowed. We illustrate these results with several examples, including winning strategies, simulation, and a declarative specification for let-polymorphism type checking [Milner 1978]. Parts of this chapter appeared in the conference papers [Miller 2007a] and [Nigam 2008a].

Chapter 4 contains some of the results related to the specification of Logical Frameworks. In this chapter, we describe how object-logic formulas, sequents and inference rules can be encoded in linear logic. Here, we also distinguish three increasing levels for adequacy: the level of relative completeness – when two proof systems prove the same theorems; the level of full completeness of proofs – when two proof systems have the same proofs; and the level of full completeness of derivations – when two systems have the same (open) derivations. Later, we construct from a single linear logic theory equivalent theories that encode with the strongest level of adequacy different proof systems, namely natural deduction (normal and non-normal), sequent calculus (with and without cut) [Gentzen 1969], natural deduction with general elimination rules [von Plato 2001], free deduction [Parigot 1992] and tableaux proof systems [D'Agostino 1994, Smullyan 1968a]. A direct consequence of the strong level of adequacy obtained and the fact that these theories are equivalent is the relative completeness of the encoded proof systems; for example, we show that sequent calculus and natural deduction systems for intuitionistic logic prove the same theorems. Parts of this chapter appeared in the conference paper [Nigam 2008b] and in its extended version [Nigam 2009b].

Chapter 5 investigates the proof system for linear logic with subexponentials called *SELL*. In particular, we review the conditions on the preorder of subexponentials needed to show that the cut-elimination theorem holds for *SELL* [Danos 1993]. Then, we proceed by proposing a focused proof system for *SELL* starting first by considering the case when there are no *relevant formulas*, that is, there are no formulas that can contract but not weaken. Later, we describe two different focused systems for the general case. Finally, we also propose a new logic that allows for the creation and modification of subexponentials. Parts of this chapter appeared in [Nigam 2009a].

Chapter 6 revisits the concerns related to Logical Frameworks, discussed in Chapter 4, exploiting the increase of expressiveness obtained by the addition of subexponentials to linear logic. In particular, we propose linear logic theories that encode different proof systems with the strongest level of adequacy, namely a *multi-conclusion system* for intuitionistic logic [Maehara 1954] and several focusing proof systems [Herbelin 1995, Dyckhoff 2006, Liang 2008].

Chapter 7 studies what algorithms can be expressed in linear logic with subexponentials. In particular, we use *subexponentials* to assign *locations* to multisets of formulas within a linear logic programming setting. Treating locations as subexponentials greatly increases the algorithmic expressiveness of logic. To illustrate this new expressiveness, we show that focused proof search can be precisely linked to a simple algorithmic specification language that contains while-loops, conditionals, and insertion into and deletion from multisets, called Bag. We illustrate this result with several graph algorithms, such as *Dijkstra's algorithm* for finding the shortest distances in a positively weighted graph and an algorithm for checking if a graph is bipartite. We also illustrate in this chapter that by changing focusing annotations, such as *delay operators*, we can capture different intended operational semantics, which can greatly affect the behavior of programs. Parts of this chapter appeared in [Nigam 2009a].

Chapter 8 concludes this thesis by summarizing its main contributions.

Chapter 2

Non-canonical aspects of the Sequent Calculus

Proof theory is the field of mathematics that investigates the properties and the structure of *formal* proofs. Although the idea of proof in mathematics is very old, its precise formalization dates back only a past hundred years or so, with Hilbert's axiomatic proof system. Since then many have contributed with different formalisms that can be used for the study of proofs. In this chapter, we introduce one such formalism, proposed by Gentzen in 1935 [Gentzen 1969], called *Sequent Calculus*, which also has deep connections with computer science. Here, we also highlight some of its *non-canonical* aspects. This is in no way an exhaustive exposition of these topics, but just a gentle introduction, focusing only on the concepts needed in the following chapters. Nevertheless, the reader can find more details in the following references [Kleene 1968, Troelstra 1996, Girard 1989].

We start by introducing the notion of *canonical forms*. Given a set of objects, $\mathcal{S}$, with an equivalence relation, we say that some objects in $\mathcal{S}$ are the canonical forms of $\mathcal{S}$ if all objects in $\mathcal{S}$ are equivalent to exactly one canonical form. Intuitively, the canonical forms of a set $\mathcal{S}$ can be seen as the representative objects of the equivalence classes specified by the given equivalence relation. In the following sections, we point out many aspects of the sequent calculus, such as first order quantifiers and different *focused proofs* obtained from different *polarity assignments* for literals, that do not have a *unique canonical form*, but distinct canonical representations for such aspects.

This chapter is structured as follows: after introducing the syntax of formulas in Section 2.1, we explain, in Section 2.2, the main vocabulary for the *Sequent Calculus*, by introducing the sequent calculus system LK for classical logic. Section 2.3 introduces the sequent calculus system LJ for intuitionistic logic and Section 2.4 introduces the sequent calculus system LL for linear logic. Finally, in Section 2.5, we describe focusing and introduce the focused proof system, LLF, for linear logic.

2.1 Syntax

We use a similar approach for syntax as in Church's type theory [Church 1940]. Instead of using a single type, i, for individuals, we allow terms to have any simple type, that does not contain the type o, which is reserved for propositions. Moreover, we generally assume that terms are in $\beta\eta$-long forms. For example, the types for the classical logic connectives are as follows:

$$\begin{aligned}&\top, \bot : o\\ &\Rightarrow, \wedge, \vee : o \to o \to o\\ &\exists_\gamma, \forall_\gamma : (\gamma \to o) \to o\end{aligned}$$

Here, renaming and substitution of variables are handled by using the standard α-conversions and β-reductions in the λ-calculus. Notice as well that in this setting the universal and the existential quantifications do not have a unique *canonical form*, as there is a pair of quantifiers for each type γ different from o. In most of the cases, it will be clear from the context which type of quantifier we are using, and therefore we will elide the subscript γ in these connectives. To ease notation, we also write quantified formulas, such as $\forall \lambda x.P$ and $\exists \lambda x.P$, as $\forall x P$ and $\exists x P$.

2.2 Sequent Calculus

Introduced by Gentzen in 1935 [Gentzen 1969], the *Sequent Calculus*, together with Natural Deduction [Gentzen 1969, Prawitz 1965], has been used as the main tool for the study of the structure of proofs. A sequent calculus system is a collection of *inference rules* that derives *sequents*. For example, the sequent calculus for classical logic LK, shown in Figure 2.1, derives sequents of the form $\Gamma \vdash \Delta$, where both Γ, denoting assumptions, and Δ, denoting propositions, are multisets of formulas. Intuitively, a sequent $\Gamma \vdash \Delta$ denotes that the conjunction of the formulas in Γ entails (denoted by the symbol $\vdash$ called *turnstyle*) the disjunction of the formulas in Δ. We use the sequent calculus LK to specify the main vocabulary used throughout this thesis.

In a sequent calculus rule, we call the sequent(s) appearing above the horizontal line its *premise(s)* and the sequent appearing below the horizontal line its *conclusion*. For example, the initial rule I has no premises, while the rule $\wedge_r$ has two premises. The *principal* formula of a rule is the formula distinguished in its conclusion, and the *active* formulas of a rule are the subformulas of the principal formula in its premises and its principal formula. For example, the active formulas of the rule $\wedge_r$ are P, Q and the principal formula $P \wedge Q$. The remaining formulas are the *side-formulas*. We often say that a formula that appears to the left (respectively right) of the turnstyle is on the left-hand-side (respectively right-hand-side) of the sequent. The rules in a sequent calculus system can be classified into three groups: (1) the *identity rules* are the rules that require to check if two formulas are the same. In LK, the axiom rule is applicable if, in its conclusion, there is a formula on the right that is the same as a formula on the left of the turnstyle. On the other hand, the cut rule is applicable only if the same formula appears in the left-hand-side of its left premise and in the right-hand-side of its right premise. Notice that the cut rule is the only rule in LK where the formulas in the premises are not subformulas of formulas in the conclusion. We shall return to this observation when we describe the *cut-elimination* theorem and the *subformula property*. (2) The *structural rules* are rules that do not operate on any logical connective, but on sequents directly. For example, the structural rules in LK just express that the formulas, appearing in the left and right side of the turnstyle, can be seen as sets of formulas. In fact, all the systems considered in this thesis have exchange rules, and, therefore, we no longer consider these rules explicitly. However, later in this chapter, we specify substructural logics that do not allow weakening and contraction rules to be applied to all formulas and those rules still remain of interest. (3) The *logical rules* are the rules that decompose logical connectives.

A *sequent calculus derivation* is a tree structure in which all of its nodes are decorated with valid sequents and for any sequent, when seen as the conclusion of a rule, and its children, when seen as the premises of a rule, constitute a valid instance of a rule in the system. The *premises* of a derivation are the sequents at the open leaves of the tree. The sequent at the root of the tree is called the *endsequent* of the derivation. A *sequent calculus proof* is a derivation that does not contain premises. For example, the object to the left is an LK proof of the *excluded middle* and the object to the right is an LK proof of one of the *De Morgan's laws*:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,[I]}{\vdash A^\perp, A}\,[\neg_r]}{\vdash A \vee A^\perp, A}\,[\vee_{r2}]}{\vdash A \vee A^\perp, A \vee A^\perp}\,[\vee_{r1}]}{\vdash A \vee A^\perp}\,[C_r] \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,[I]}{\vdash A^\perp, A}\,[\neg_r]}{\vdash A^\perp \vee B^\perp, A}\,[\vee_{r1}] \quad \dfrac{\dfrac{\dfrac{}{B \vdash B}\,[I]}{\vdash B^\perp, B}\,[\neg_r]}{\vdash A^\perp \vee B^\perp, B}\,[\vee_{r2}]}{\vdash A \wedge B, A^\perp \vee B^\perp}\,[\wedge_r]}{(A \wedge B)^\perp \vdash A^\perp \vee B^\perp}\,[\neg_l]}{\vdash (A \wedge B)^\perp \Rightarrow (A^\perp \vee B^\perp)}\,[\Rightarrow_r]$$

In fact, because of LK's symmetry, all De Morgan's laws, listed below, are provable in LK:

- $(P \wedge Q)^{\perp} \equiv P^{\perp} \vee Q^{\perp}$;
- $(P \vee Q)^{\perp} \equiv P^{\perp} \wedge Q^{\perp}$;
- $(\exists x.P)^{\perp} \equiv \forall x.P^{\perp}$;
- $(\forall x.P)^{\perp} \equiv \exists x.P^{\perp}$;
- $(A \Rightarrow B)^{\perp} \equiv A \wedge B^{\perp}$;

where $P \equiv Q$ is an abbreviation for the sequent $\vdash P \Rightarrow Q \wedge Q \Rightarrow P$. These laws allow the use of a more concise presentation for LK, where sequents are *one sided*, that is, sequents where formulas appear only in one side of the turnstyle. Intuitively, we replace a two sided sequent $\Gamma \vdash \Delta$ by the one sided sequent $\vdash \Gamma^{\perp}, \Delta$. We consider the negation of a formula as its *negation normal form* obtained by pushing the negation inside the formula, via De Morgan's laws, until negations only appear before atoms. Moreover, for atoms, A, we rewrite $A^{\perp\perp}$ as A. The one sided system for LK is depicted in Figure 2.2. Notice that not all systems have a one sided version; the most notorious example being the proof system LJ for *intuitionistic logic*, where sequents have at most one formula in the right-hand-side of the turnstyle. When possible, we prefer to use the one-sided version of a system because of its simplicity.

In most of the sequent calculus systems, the initial rule can be restricted to only atomic principal formulas without the loss of *completeness*, that is, being able to prove the same set of theorems.

Proposition 2.1 *A formula F is provable in LK if and only if it is provable in LK by using only instances of atomic initial rules.*

We can reduce instances of non-atomic initial rules to instances of atomic initial rules by repeatedly applying transformations of the form:

$$\frac{}{P \wedge Q \vdash P \wedge Q}\,[I] \quad \rightsquigarrow \quad \frac{\dfrac{\dfrac{}{P \vdash P}\,[I]}{P \wedge Q \vdash P}\,[\wedge_l] \quad \dfrac{\dfrac{}{Q \vdash Q}\,[I]}{P \wedge Q \vdash Q}\,[\wedge_l]}{P \wedge Q \vdash P \wedge Q}\,[\wedge_r]$$

We now return to the cut rule. The cut rule differs from all others rules in LK because it introduces in the proof a new formula, called *cut formula*. This formula can be seen as an auxiliary lemma that when introduced in a proof, has to be proved in the left premise of a cut rule and can be used in the proof of the right premise. As the cut formula does not appear in the conclusion of the cut rule, there is some ingenuity involved to discover which cut formula to use. However, Gentzen showed that any proof containing cuts can be transformed into a proof that does not contain cuts, called *cut-free*. This result is called the *Hauptsatz* and gave birth to the field of *Structural Proof Theory*. The proof of this theorem can be found in many references, *e.g.* in Chapter 4 of Troelstra and Schwichtenberg's book [Troelstra 1996].

Theorem 2.2 *The cut rule is admissible in LK: any proof containing cuts can be transformed into a cut-free proof with the same end-sequent.*

Two important consequences of the cut-elimination theorem are the *subformula property* and the *consistency* of the logic. The former consequence is stated as follows:

Proposition 2.3 *All formulas appearing in a cut-free proof are subformulas of formulas appearing at its endsequent.*

The latter consequence states that it is not possible to prove both a formula and its negation.

Proposition 2.4 *For any formula P, it is not the case that both sequents $\vdash P$ and $\vdash P^{\perp}$ are provable in LK.*

IDENTITY RULES

$$\frac{}{P \vdash P}\ [I] \qquad \frac{\Gamma_1 \vdash P, \Delta_1 \quad \Gamma_2, P \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\ [Cut]$$

LOGICAL RULES

$$\frac{}{\bot, \Gamma \vdash \Delta}\ [\bot] \qquad \frac{}{\Gamma \vdash \Delta, \top}\ [\top]$$

$$\frac{\Gamma \vdash \Delta, P}{\Gamma, P^{\bot} \vdash \Delta}\ [\neg_l] \qquad \frac{\Gamma, P \vdash \Delta}{\Gamma \vdash P^{\bot}, \Delta}\ [\neg_r]$$

$$\frac{P_i, \Gamma \vdash \Delta}{P_1 \wedge P_2, \Gamma \vdash \Delta}\ [\wedge_{li}] \qquad \frac{\Gamma \vdash P, \Delta \quad \Gamma \vdash Q, \Delta}{\Gamma \vdash P \wedge Q, \Delta}\ [\wedge_r]$$

$$\frac{P, \Gamma \vdash \Delta \quad Q, \Gamma \vdash \Delta}{P \vee Q, \Gamma \vdash \Delta}\ [\vee_l] \qquad \frac{\Gamma \vdash P_i, \Delta}{\Gamma, \vdash P_1 \vee P_2, \Delta}\ [\vee_{ri}]$$

$$\frac{\Gamma_1 \vdash P, \Delta_1 \quad Q, \Gamma_2 \vdash \Delta_2}{P \Rightarrow Q, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\ [\Rightarrow_l] \qquad \frac{\Gamma, P \vdash Q, \Delta}{\Gamma \vdash P \Rightarrow Q, \Delta}\ [\Rightarrow_r]$$

$$\frac{P[c/x], \Gamma \vdash \Delta}{\exists x.P, \Gamma \vdash \Delta}\ [\exists_l] \qquad \frac{\Gamma \vdash P[t/x], \Delta}{\Gamma \vdash \exists x.P, \Delta}\ [\exists_r]$$

$$\frac{P[t/x], \Gamma \vdash \Delta}{\forall x.P, \Gamma \vdash \Delta}\ [\forall_l] \qquad \frac{\Gamma \vdash P[c/x], \Delta}{\Gamma \vdash \forall x.P, \Delta}\ [\forall_r]$$

STRUCTURAL RULES

$$\frac{P, P, \Gamma \vdash \Delta}{P, \Gamma \vdash \Delta}\ [C_l] \qquad \frac{\Gamma \vdash P, P, \Delta}{\Gamma \vdash P, \Delta}\ [C_r]$$

$$\frac{\Gamma \vdash \Delta}{P, \Gamma \vdash \Delta}\ [W_l] \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash P, \Delta}\ [W_r]$$

$$\frac{\Gamma, Q, P, \Gamma' \vdash \Delta}{\Gamma, P, Q, \Gamma' \vdash \Delta}\ [E_l] \qquad \frac{\Gamma \vdash \Delta, Q, P, \Delta'}{\Gamma \vdash \Delta, P, Q, \Delta'}\ [E_r]$$

Figure 2.1: The identity, logical and structural rules of the two sided presentation for LK, a sequent calculus system for classical logic. In the rules $\forall_r$ and $\exists_l$, the eigenvariable c does not appear free in Γ nor Δ.

IDENTITY RULES

$$\frac{}{\vdash P, P^{\perp}}\ [I] \qquad \frac{\vdash P, \Delta_1 \quad \vdash P^{\perp}, \Delta_2}{\vdash \Delta_1, \Delta_2}\ [Cut]$$

LOGICAL RULES

$$\frac{}{\vdash \Delta, \top}\ [\top]$$

$$\frac{\vdash P_i, \Delta}{\vdash P_1 \vee P_2, \Delta}\ [\vee_i] \qquad \frac{\vdash P, \Delta \quad \vdash Q, \Delta}{\vdash P \wedge Q, \Delta}\ [\wedge]$$

$$\frac{\vdash P^{\perp}, Q, \Delta}{\vdash P \Rightarrow Q, \Delta}\ [\Rightarrow]$$

$$\frac{\vdash P[t/x], \Delta}{\vdash \exists x.P, \Delta}\ [\exists] \qquad \frac{\vdash P[c/x], \Delta}{\vdash \forall x.P, \Delta}\ [\forall]$$

STRUCTURAL RULES

$$\frac{\vdash P, P, \Delta}{\vdash P, \Delta}\ [C] \qquad \frac{\vdash \Delta}{\vdash P, \Delta}\ [W]$$

Figure 2.2: The identity, logical and structural rules of the one sided presentation for LK, a sequent calculus system for classical logic. In the rule $\forall$, the eigenvariable c does not appear free in Δ.

Proof Assume by contradiction that a formula, P, and its negation, $P^{\perp}$, are both provable in LK. Then the empty sequent could be proved by using two consecutive cuts:

$$\dfrac{\dfrac{\Xi_1}{\vdash P^{\perp}}\ [Cut] \qquad \dfrac{\dfrac{\Xi_2}{\vdash P}\ [Cut] \qquad \dfrac{}{\vdash P, P^{\perp}}\ [I]}{\vdash P}\ [Cut]}{\vdash}\ [Cut]$$

However, as there is clearly no cut-free proof for the empty sequent, this leads to a contradiction. □

The cut-elimination theorem has also important connections with computer science, namely it is connected to two programming paradigms: *functional programming* and *logic programming*. The former connection is the *Curry-Howard* isomorphism that establishes the correspondence between proof systems and models of computation, such as λ-calculus: a program corresponds to a proof and the formula a proof proves corresponds to the type of the program. Proofs are run by performing a step of the cut-elimination procedure, which in λ-calculus corresponds to β-reduction. The *canonical proofs* for this paradigm are the cut-free proofs obtained by applying the cut-elimination algorithm. The latter connection comes from the *computation-as-proof-search* point of view. Sets of logic formulas represent logic programs, sequents represent states of the *world*, and cut-free sequent calculus proofs represent computation traces. The idea is that, when a clause in a logic program is used (from bottom-up), the conclusion of the derivation corresponds to the initial state of the world, and its premises to the states obtained after a computation step is performed. We discuss in Section 2.5 the *canonical proofs* for this paradigm, called *focused proofs*. In this thesis, we concentrate mainly on the latter paradigm.

2.3 Intuitionistic Logic

The main difference between intuitionistic logic and classical logic is that the former only allows for constructive proofs, that is, proofs that also demonstrate how to construct the objects in the proof. Consider, for example, the following classical logic non-constructive proof of the following theorem:

there exist two irrational numbers, a and b, such that a^b is rational.

We know that $\sqrt{2}$ is irrational. Consider the number $\sqrt{2}^{\sqrt{2}}$: either it is rational or irrational. In the former case, it would mean that there are two numbers $a = b = \sqrt{2}$ for which a^b is rational. For the latter case, if $\sqrt{2}^{\sqrt{2}}$ is irrational, we show that if $a = \sqrt{2}^{\sqrt{2}}$ and $b = \sqrt{2}$ then a^b is rational: $a^b = \sqrt{2}^{(\sqrt{2}\times\sqrt{2})} = \sqrt{2}^2 = 2$, proving the theorem.

This proof would not be valid in intuitionistic logic as one cannot construct from this proof two irrational numbers a and b such that a^b is rational. Its non-constructiveness comes from the use of the law of excluded-middle $\forall A.A \vee A^\perp$ (in the proof, the cases *either it is rational or irrational*), which is not true in intuitionistic logic.

The proof system for intuitionistic logic LJ is shown in Figure 2.3, where we define intuitionistic negation $\neg A$ as $A \supset \perp$. Differently from classical logic, the right-hand-side of sequents is allowed to contain at most one formula. Because of this restriction, the De Morgan's laws are not provable in intuitionistic logic, and, therefore, there is no one-sided presentation for LJ. Moreover, as the cut-elimination theorem holds for intuitionistic logic, LJ is consistent and admits the subformula property.

In the two last sections, we gave a very brief introduction to the sequent calculus for classical and intuitionistic logics. Later in Chapters 4 and 6, we show and discuss other sequent calculus systems for these logics. Now we proceed to *linear logic*, a substructural logic introduced by Girard [Girard 1987], where differently from classical logic weakening and contraction cannot be applied to all formulas.

2.4 Linear Logic

In a classical or intuitionistic logic proof, because of the contraction and the weakening rules, hypotheses can be used as many times as necessary or even not be used at all. In linear logic the story is different. As contraction and weakening cannot be applied to all formulas, some hypotheses must be used and be used only once.

The syntax for linear logic formulas is given below and its rules are depicted in Figure 2.4:

$$\begin{array}{rcl} P & ::= & A \mid P \otimes P \mid P \oplus P \mid 1 \mid 0 \mid !P \mid \exists x.P \mid \\ & & A^\perp \mid P \,\&\, P \mid P \parr P \mid \perp \mid \top \mid ?P \mid \forall x.P \end{array}$$

Weakening and contraction can only be applied to formulas whose main connective is a question mark. This connective acts like a guard keeper: seeing from bottom up, the ? allows structural rules to be applied to formulas, and when enough copies of a formula are made, the question mark can be removed by using the *dereliction rule* $D?$. We classify the formulas that do not allow neither contraction nor weakening as *linear* and the formulas that allow contraction and weakening as *unbounded*. Because of this control over structural rules in linear logic, disjunction and conjunction appear in two different forms: the multiplicative ones, $\otimes$ (called *tensor*) and $\parr$ (called *par*), and the additive ones, $\&$ (called *with*) and $\oplus$ (called *plus*). This was not the case in classical logic because one type of conjunction (respectively disjunction) could be derived from the other. The following derivations illustrate how

IDENTITY RULES

$$\frac{}{P \longrightarrow P}\ [I] \qquad \frac{\Gamma_1 \longrightarrow P \quad \Gamma_2, P \longrightarrow C}{\Gamma_1, \Gamma_2 \longrightarrow C}\ [Cut]$$

LOGICAL RULES

$$\frac{}{\bot, \Gamma \longrightarrow .}\ [\bot] \qquad \frac{}{\Gamma \longrightarrow t}\ [t_r]$$

$$\frac{P_i, \Gamma \longrightarrow C}{P_1 \wedge P_2, \Gamma \longrightarrow C}\ [\wedge_{li}] \qquad \frac{\Gamma \longrightarrow P \quad \Gamma \longrightarrow Q}{\Gamma \longrightarrow P \wedge Q}\ [\wedge_r]$$

$$\frac{P, \Gamma \longrightarrow C \quad Q, \Gamma \longrightarrow C}{P \vee Q, \Gamma \longrightarrow C}\ [\vee_l] \qquad \frac{\Gamma \longrightarrow P_i}{\Gamma \longrightarrow P_1 \vee P_2}\ [\vee_{ri}]$$

$$\frac{\Gamma_1 \longrightarrow P \quad Q, \Gamma_2 \longrightarrow C}{P \supset Q, \Gamma_1, \Gamma_2 \longrightarrow C}\ [\supset_l] \qquad \frac{\Gamma, P \longrightarrow Q}{\Gamma \longrightarrow P \supset Q}\ [\supset_r]$$

$$\frac{P[c/x], \Gamma \longrightarrow C}{\exists x.P, \Gamma \longrightarrow C}\ [\exists_l] \qquad \frac{\Gamma \longrightarrow P[t/x]}{\Gamma \longrightarrow \exists x.P}\ [\exists_r]$$

$$\frac{P[t/x], \Gamma \longrightarrow C}{\forall x.P, \Gamma \longrightarrow C}\ [\forall_l] \qquad \frac{\Gamma \longrightarrow P[c/x]}{\Gamma \longrightarrow \forall x.P}\ [\forall_r]$$

STRUCTURAL RULES

$$\frac{P, P, \Gamma \longrightarrow C}{P, \Gamma \longrightarrow C}\ [C_l]$$

$$\frac{\Gamma \longrightarrow C}{P, \Gamma \longrightarrow C}\ [W_l] \qquad \frac{\Gamma \longrightarrow .}{\Gamma \longrightarrow P}\ [W_r]$$

Figure 2.3: The identity, logical and structural rules for LJ, a sequent calculus system for intuitionistic logic. In the rules $\forall_r$ and $\exists_l$, the eigenvariable c does not appear free in Γ nor C, and C denotes either a formula or no formula.

Identity Rules

$$\frac{}{\vdash P, P^{\perp}}\ [I] \qquad \frac{\vdash \Gamma, P^{\perp} \quad \vdash \Delta, P}{\vdash \Gamma, \Delta}\ [Cut]$$

Logical Rules

$$\frac{}{\vdash 1}\ [1] \qquad \frac{}{\vdash \Gamma, \top}\ [\top] \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \perp}\ [\perp]$$

$$\frac{\vdash \Gamma, P_i}{\vdash \Gamma, P_1 \oplus P_2}\ [\oplus_i] \qquad \frac{\vdash \Gamma, P \quad \vdash \Gamma, Q}{\vdash \Gamma, P \,\&\, Q}\ [\&]$$

$$\frac{\vdash \Gamma, P, Q}{\vdash \Gamma, P \parr Q}\ [\parr] \qquad \frac{\vdash \Gamma, P \quad \vdash \Delta, Q}{\vdash \Gamma, \Delta, P \otimes Q}\ [\otimes]$$

$$\frac{\vdash \Gamma, P[t/x]}{\vdash \Gamma, \exists x.P}\ [\exists] \qquad \frac{\vdash \Gamma, P[c/x]}{\vdash \Gamma, \forall x.P}\ [\forall]$$

$$\frac{\vdash \Gamma, P}{\vdash \Gamma, ?P}\ [D?] \qquad \frac{\vdash ?\Gamma, P}{\vdash ?\Gamma, !\, P}\ [!]$$

Structural Rules

$$\frac{\vdash \Gamma, ?P, ?P}{\vdash \Gamma, ?P}\ [C] \qquad \frac{\vdash \Gamma}{\vdash \Gamma, ?P}\ [W]$$

Figure 2.4: Rules for the one sided version of linear logic. In the rule $\forall$, the eigenvariable c does not appear free in Γ.

to derive in LK the additive conjunction, $\wedge_a$, from the multiplicative one, $\wedge_m$.

$$\frac{\vdash \Gamma, \Delta, P \quad \vdash \Gamma, \Delta, Q}{\vdash \Gamma, \Delta, P \wedge_a Q}\ [\wedge_a] \qquad \rightsquigarrow \qquad \frac{\dfrac{\vdash \Gamma, \Delta, P \quad \vdash \Gamma, \Delta, Q}{\vdash \Gamma, \Gamma, \Delta, \Delta, P \wedge_m Q}\ [\wedge_m]}{\vdash \Gamma, \Delta, P \wedge_m Q}\ [n \times C]$$

Similarly, there are *different* connectives for truth and false, called units: $\top$ and 1 for truth and $\perp$ and 0 for false.

As the reader might have already noticed, there is a one sided version for linear logic, since there are linear logic proofs for the De Morgan's laws depicted below. Hence, we can push negation inside a formula until we obtain a formula in negation normal form.

- $(P \otimes Q)^{\perp} \equiv P^{\perp} \parr Q^{\perp}$;
- $(P \parr Q)^{\perp} \equiv P^{\perp} \otimes Q^{\perp}$;
- $(P \,\&\, Q)^{\perp} \equiv P^{\perp} \oplus Q^{\perp}$;
- $(P \oplus Q)^{\perp} \equiv P^{\perp} \,\&\, Q^{\perp}$;
- $(\exists x.P)^{\perp} \equiv \forall x.P^{\perp}$;
- $(\forall x.P)^{\perp} \equiv \exists x.P^{\perp}$;
- $(?P)^{\perp} \equiv !\, P^{\perp}$;
- $(!\, P)^{\perp} \equiv ?P^{\perp}$.

where the linear equivalence $P \equiv Q$ denotes the linear logic sequent $\vdash (P \multimap Q) \otimes (Q \multimap P)$, $\multimap$ is the linear implication and $A \multimap B$ denotes the linear logic formula $A^{\perp} \parr B$. These equivalences also illustrate the dualities in the logic: the pairs of connectives $(\otimes, \parr)$, $(\oplus, \&)$, and $(?, !)$ are duals. For the units the equivalences $\top^{\perp} \equiv 0$ and $1^{\perp} \equiv \perp$ are also provable.

The connectives ? (called question-mark) and ! (called bang) are, commonly, called *exponentials* because of the equivalences below, that are provable in linear logic and look like the equality: $e^{x+y} =$

$e^x \times e^y$.

- $!(P \,\&\, Q) \equiv (!\,P) \otimes (!\,Q)$;
- $?(P \oplus Q) \equiv (?P) \parr (?Q)$.

An important difference between the exponentials and the remaining connectives is that while the latter have canonical forms, for example the tensor is canonical, the former do not have canonical representations. Consider two different types of exponentials, say, one colored as *blue*, $?^b$ and $!^b$, and the other as *red*, $?^r$ and $!^r$. Moreover, consider that their introduction rules, called *dereliction rule* for question-marks and *promotion rule* for bangs, are specified as follows :

$$\frac{\vdash ?^b\Gamma, F}{\vdash ?^b\Gamma, !^b F}\,[!^b] \qquad \frac{\vdash \Gamma, F}{\vdash \Gamma, ?^b F}\,[D?^b] \qquad \frac{\vdash ?^r\Gamma, F}{\vdash ?^r\Gamma, !^r F}\,[!^r] \qquad \frac{\vdash \Gamma, F}{\vdash \Gamma, ?^r F}\,[D?^r]$$

It is easy to check that the formula $!^b F \equiv !^r F$ is not provable in this system for any formula F, thus the two types of exponentials are not canonical. In fact, there are infinitely many distinct exponentials. This was investigated by Danos *et al.* in [Danos 1993], where they propose a system with possibly infinitely many colors for ? and !. We return to this system in Chapter 5, when we propose a *focused version* for it. We then exploit this non-canonical aspect of linear logic in Chapters 6 and 7.

Girard proved that the cut-elimination theorem also holds for linear logic:

Theorem 2.5 *The cut elimination theorem holds for linear logic.*

Andreoli showed [Andreoli 1992] that the *dyadic* representation for linear logic, shown in Figure 2.5, is sound and complete with respect to linear logic. Dyadic sequents are of the form $\vdash \Theta : \Gamma$, which can be read as the linear logic sequent $\vdash ?\Theta, \Gamma$. This alternative representation simplifies the handling of exponentials and structural rules. Unbounded formulas are contracted before a tensor and a cut rule and weakened before an initial rule. We often refer to the context Θ as the *unbounded context* and the context Γ as the *linear* or *bounded context*. We return to this representation in the next section.

Form the computation-as-proof-search paradigm, linear logic can be seen as the *logic of resources*. Linear logic sequents represent states of the world; the linear formulas present in the sequent represent the number of resources available; and linear logic proofs represent computations where the resources available might change. For example, in the sequent $\vdash ?(\mathit{euro}^\perp \otimes \mathit{coffee}), \mathit{euro}$, the linear atom specifies that an agent has only one euro, and the unbounded formula specifies the action of exchanging a euro for a coffee. Hence, this agent could *consume* its *euro* and obtain a *coffee*, represented by the sequent $\vdash ?(\mathit{euro}^\perp \otimes \mathit{coffee}), \mathit{coffee}$. From a proof theoretic perspective, linear logic is used as the *logic behind logics* to understand and study other logics, such as classical and intuitionistic logics. For example, Girard showed that one could capture intuitionistic logic in linear logic by using the following translation from intuitionistic formulas to linear logic formulas [Girard 1987], where A is an atom:

$$\begin{array}{rclcrcl}
\ulcorner P \wedge Q \urcorner & \equiv & \ulcorner P \urcorner \,\&\, \ulcorner Q \urcorner & \qquad & \ulcorner P \vee Q \urcorner & \equiv & !\ulcorner P \urcorner \oplus !\ulcorner Q \urcorner \\
\ulcorner \top \urcorner & \equiv & \top & & \ulcorner \bot \urcorner & \equiv & 0 \\
\ulcorner P \supset Q \urcorner & \equiv & ?\ulcorner P \urcorner^\perp \parr \ulcorner Q \urcorner & & \ulcorner A \urcorner & \equiv & A \\
\ulcorner \exists x.P \urcorner & \equiv & \exists x.\ulcorner P \urcorner & & \ulcorner \forall x.P \urcorner & \equiv & \forall x.\ulcorner P \urcorner
\end{array}$$

We explore the proof theoretic perspective in Chapters 4 and 6 and the computer science perspective in Chapter 7.

We continue, in the next section, with one of the cornerstones of this thesis: *focusing*.

2.5 Focusing

The systems that we have considered so far do not have a strong proof search discipline. Proofs are constructed in a *small step* fashion: one applies any applicable rule until no open leaves remain.

Identity Rules

$$\frac{}{\vdash \Theta : P, P^{\perp}}\ [I] \qquad \frac{\vdash \Theta : \Gamma, P^{\perp} \quad \vdash \Theta : \Delta, P}{\vdash \Theta : \Gamma, \Delta}\ [Cut]$$

Logical Rules

$$\frac{}{\vdash \Theta : 1}\ [1] \qquad \frac{}{\vdash \Theta : \Gamma, \top}\ [\top] \qquad \frac{\vdash \Theta : \Gamma}{\vdash \Theta : \Gamma, \perp}\ [\perp]$$

$$\frac{\vdash \Theta : \Gamma, P_i}{\vdash \Theta : \Gamma, P_1 \oplus P_2}\ [\oplus_i] \qquad \frac{\vdash \Theta : \Gamma, P \quad \vdash \Theta : \Gamma, Q}{\vdash \Theta : \Gamma, P \,\&\, Q}\ [\&]$$

$$\frac{\vdash \Theta : \Gamma, P, Q}{\vdash \Theta : \Gamma, P \parr Q}\ [\parr] \qquad \frac{\vdash \Theta : \Gamma, P \quad \vdash \Theta : \Delta, Q}{\vdash \Theta : \Gamma, \Delta, P \otimes Q}\ [\otimes]$$

$$\frac{\vdash \Theta : \Gamma, P[t/x]}{\vdash \Theta : \Gamma, \exists x.P}\ [\exists] \qquad \frac{\vdash \Theta : \Gamma, P[c/x]}{\vdash \Theta : \Gamma, \forall x.P}\ [\forall]$$

$$\frac{\vdash \Theta, P : \Gamma, P}{\vdash \Theta, P : \Gamma}\ [D?] \qquad \frac{\vdash \Theta : P}{\vdash \Theta : !\, P}\ [!]$$

Structural Rules

$$\frac{\vdash \Theta, P : \Gamma}{\vdash \Theta : \Gamma, ?P}\ [?]$$

Figure 2.5: Rules for the dyadic version of linear logic. In the rule $\forall$, the eigenvariable c does not appear free in Γ.

The problem of this weak discipline is that it has a great deal of non-determinism in proof search, as one can choose any formula in the sequent that a rule can be applied to. Thus, there are too many proofs available in such systems to express, from the computation-as-proof-search perspective, relevant computational behaviors. We need *canonical* proofs that have a better proof search behavior. Andreoli provided such proofs by introducing the focusing discipline [Andreoli 1992] where proofs are constructed, instead, in a *big step* fashion.

Andreoli proved the completeness of the focused proof system for linear logic, LLF, given in Figure 2.6. Focusing proof systems involve applying inference rules in alternating phases. For this we first classify connectives and formulas as synchronous and asynchronous, as follows:

Definition 2.6 The connectives $\parr, \&, ?, \top, \bot$ and $\forall$ (respectively $\otimes, \oplus, 1, 0, !$ and $\exists$) are classified as *asynchronous* (respectively *synchronous*). Formulas whose main connective is asynchronous (respectively synchronous) are classified as asynchronous (respectively synchronous). Inference rules that introduce an asynchronous (respectively synchronous) connective are classified as asynchronous (respectively synchronous).

This classification is rather natural in the sense that all right introduction rules for asynchronous formulas are invertible, while such introduction rules for synchronous formulas are not necessarily invertible. However, this classification does not apply well to literals. We use, instead, the adjectives *positive* and *negative* polarity. Moreover, we say that a formula is positive if it is synchronous or a positive literal, and negative if it is asynchronous or a negative literal. In the asynchronous phase, composed by sequents in *tryadic* form $\vdash \Theta : \Gamma \Uparrow L$, rules are applied only to negative formulas appearing in the list of formulas L, while positive formulas are moved to one of the multisets, Θ or Γ, on the left of the $\Uparrow$, by using the $R\Uparrow$ or ? rules. When L is empty, the synchronous phase begins by using one of the decide rules D_1 or D_2 to select a single formula on which to "focus": the judgment $\vdash \Theta : \Gamma \Downarrow F$ denotes such a sequent which is focused on F. Rules are then applied hereditarily to subformulas of F until a negative subformula is encountered, at which time, the release rule $R\Downarrow$ is used and another asynchronous phase begins.

We write $\vdash_{llf} \Theta : \Gamma \Uparrow$ to indicate that the sequent $\vdash \Theta : \Gamma \Uparrow$ has a proof in LLF; $\vdash_{llf} \Theta : \Gamma \Downarrow$ to indicate that the sequent $\vdash \Theta : \Gamma \Downarrow$ has a proof in LLF; and $\vdash_{ll} \Gamma$ to indicate that the sequent $\vdash \Gamma$ is provable in linear logic.

The following proposition can be proved by a simple induction on the structure of focused proofs.

Proposition 2.7 *Let Θ, Γ, and Δ be multisets of formulas and let L be a list of formulas and F a formula. If $\vdash \Theta : \Gamma \Uparrow L$ has a proof then $\vdash \Theta, \Delta : \Gamma \Uparrow L$ has a proof of the same height. If $\vdash \Theta : \Gamma \Downarrow F$ has a proof then $\vdash \Theta, \Delta : \Gamma \Downarrow F$ has a proof of the same height.*

The two-phase structure of LLF proofs allows us to collect introduction rules into "macro-rules" that can be seen as introducing "synthetic connectives". For example, if the formulas A_1, A_2, A_3 are negative formulas, then we can view the positive formula $A_1 \oplus (A_2 \otimes A_3)$ as a synthetic connective with the following two "macro-rules" below:

$$\frac{\vdash \Theta : \Gamma \Uparrow A_1}{\vdash \Theta : \Gamma \Downarrow A_1 \oplus (A_2 \otimes A_3)} \qquad \frac{\vdash \Theta : \Gamma_1 \Uparrow A_2 \quad \vdash \Theta : \Gamma_2 \Uparrow A_3}{\vdash \Theta : \Gamma_1, \Gamma_2 \Downarrow A_1 \oplus (A_2 \otimes A_3)}$$

That is, within the LLF proof system, there are only these two ways to conclude a sequent focused on this formula without the possibility to interleave other introduction rules ("micro-rules") with those that comprise these two macro rules. Furthermore, we can compose a synchronous phase and the following asynchronous phase to build larger connectives called *bipoles*. In the example above, if A_1 is the formula $(A \otimes B) \parr (\exists x\, C)$, then we can replace the "macro-rule" to the left by the extended "macro-rule", introducing a bipole:

$$\frac{\vdash \Theta : \Gamma, A \otimes B, \exists x\, C \Uparrow}{\vdash \Theta : \Gamma \Downarrow [(A \otimes B) \parr (\exists x\, C)] \oplus (A_2 \otimes A_3)}$$

Asynchronous Phase

$$\frac{\vdash \Theta : \Gamma \Uparrow L}{\vdash \Theta : \Gamma \Uparrow L, \bot}\ [\bot] \qquad \frac{}{\vdash \Theta : \Gamma \Uparrow L, \top}\ [\top]$$

$$\frac{\vdash \Theta : \Gamma \Uparrow L, F, G}{\vdash \Theta : \Gamma \Uparrow L, F \parr G}\ [\parr] \qquad \frac{\vdash \Theta, F : \Gamma \Uparrow L}{\vdash \Theta : \Gamma \Uparrow L, ?F}\ [?]$$

$$\frac{\vdash \Theta : \Gamma \Uparrow L, F \quad \vdash \Theta : \Gamma \Uparrow L, G}{\vdash \Theta : \Gamma \Uparrow L, F \,\&\, G}\ [\&] \qquad \frac{\vdash \Theta : \Gamma \Uparrow L, F[c/x]}{\vdash \Theta : \Gamma \Uparrow L, \forall x\, F}\ [\forall]$$

Synchronous Phase

$$\frac{}{\vdash \Theta :\Downarrow 1}\ [1] \qquad \frac{\vdash \Theta : \Gamma \Downarrow F \quad \vdash \Theta : \Gamma' \Downarrow G}{\vdash \Theta : \Gamma, \Gamma' \Downarrow F \otimes G}\ [\otimes] \qquad \frac{\vdash \Theta :\Uparrow F}{\vdash \Theta :\Downarrow\, !F}\ [!]$$

$$\frac{\vdash \Theta : \Gamma \Downarrow P_i}{\vdash \Theta : \Gamma \Downarrow P_1 \oplus P_2}\ [\oplus_i] \qquad \frac{\vdash \Theta, F : \Gamma \Downarrow F[t/x]}{\vdash \Theta : \Gamma \Downarrow \exists x\, F}\ [\exists]$$

Reaction, Identity, and Decide rules

$$\frac{}{\vdash \Theta : A_p^{\perp} \Downarrow A_p}\ [I_1] \qquad \frac{}{\vdash \Theta, A_p^{\perp} :\Downarrow A_p}\ [I_2] \qquad \frac{\vdash \Theta : \Gamma, S \Uparrow L}{\vdash \Theta : \Gamma \Uparrow L, S}\ [R\Uparrow]$$

$$\frac{\vdash \Theta : \Gamma \Downarrow P}{\vdash \Theta : \Gamma, P \Uparrow}\ [D_1] \qquad \frac{\vdash \Theta, P : \Gamma \Downarrow P}{\vdash \Theta, P : \Gamma \Uparrow}\ [D_2] \qquad \frac{\vdash \Theta : \Gamma \Uparrow N}{\vdash \Theta : \Gamma \Downarrow N}\ [R\Downarrow]$$

Figure 2.6: The focused proof system for linear logic [Andreoli 1992]. Here, L is a list of formulas, Θ is a multiset of formulas, Γ is a multiset of literals and positive formulas, $i \in \{1, 2\}$, A_p is a positive literal, N is a negative formula, P is not a negative literal,and S is a positive formula or a literal.

Andreoli [Andreoli 1992] proved the focusing theorem: for any arbitrary (*global*) assignment of polarity to literals, a formula is provable in linear logic if and only if it is provable in LLF. Andreoli considered only global assignments of polarity, that is, if a literal A is assigned positive polarity then all occurrences of A are positive and all occurrences of $A^{\perp}$ are negative. We do not show Andreoli's proof but a more enlightening and modular proof due to Miller & Saurin [Miller 2007b], showing that any linear logic proof can be transformed into a focused one by *permuting* inference rules. We are going to use this modular proof in Chapter 5 to show the completeness of other focused systems.

Theorem 2.8 *Let F be a linear logic formula. Then, $\vdash F$ is provable in linear logic iff $\vdash \cdot : \cdot \Uparrow F$ is provable in LLF.*

Proof

To prove completeness of the focused system, we consider the dyadic version of linear logic and use the method introduced by Miller and Saurin [Miller 2007b], based on *permutation lemmas* and *focalisation graphs*, for which we will need some auxiliary definitions and lemmas.

Definition 2.9 Let α and β be two rules and let $\mathcal{S}$ be a sequent that could be the conclusion of α or β. We say that α *permutes* over β, denoted as α/β, if, whenever there is a proof of $\mathcal{S}$ where α is the last rule, there is proof of $\mathcal{S}$ where β is the last rule.

Lemma 2.10 *Let α be an inference rule and β be an asynchronous rule. Then α/β.*

Proof This is clear from the invertibility of the asynchronous rules. We only show the case for $\otimes/\&$.

$$\dfrac{\dfrac{\vdash \Theta : \Gamma, F, A \quad \vdash \Theta : \Gamma, F, B}{\vdash \Theta : \Gamma, F, A \,\&\, B}\,[\&] \quad \vdash \Theta : \Delta, G}{\vdash \Theta : \Gamma, \Delta, F \otimes G, A \,\&\, B}\,[\otimes]$$

The $\otimes$ rule permutes over the $\&$ rule as follows:

$$\dfrac{\dfrac{\vdash \Theta : \Gamma, F, A \quad \vdash \Theta : \Delta, G}{\vdash \Theta : \Gamma, \Delta, F \otimes G, A}\,[\otimes] \quad \dfrac{\vdash \Theta : \Gamma, F, B \quad \vdash \Theta : \Delta, G}{\vdash \Theta : \Gamma, \Delta, F \otimes G, B}\,[\otimes]}{\vdash \Theta : \Gamma, \Delta, F \otimes G, A \,\&\, B}\,[\&]$$

□

Lemma 2.11 *If α and β are synchronous rules then α/β.*

Proof This is also a standard proof of permutations of synchronous rules. We show some of the cases, and we invite the reader to Chapter 9 of Saurin's PhD thesis [Saurin 2008] for the remaining cases.

- $(\oplus/\otimes)$:

$$\dfrac{\dfrac{\vdash \Theta : A, F, \Gamma_1 \quad \vdash \Theta : B, \Gamma_2}{\vdash \Theta : A \otimes B, F, \Gamma_1, \Gamma_2}\,[\otimes]}{\vdash \Theta : A \otimes B, F \oplus G, \Gamma_1, \Gamma_2}\,[\oplus_1] \quad \rightsquigarrow \quad \dfrac{\dfrac{\vdash \Theta : A, F, \Gamma_1}{\vdash \Theta : A, F \oplus G, \Gamma_1}\,[\oplus_1] \quad \vdash \Theta : B, \Gamma_2}{\vdash \Theta : A \otimes B, F \oplus G, \Gamma_1, \Gamma_2}\,[\otimes]$$

- $(\otimes/\exists)$:

$$\dfrac{\dfrac{\vdash \Theta : A, F[t/x], \Gamma_1}{\vdash \Theta : A, \exists x.F, \Gamma_1}\,[\exists] \quad \vdash \Theta : B, \Gamma_2}{\vdash \Theta : A \otimes B, \exists x.F, \Gamma_1, \Gamma_2}\,[\otimes] \quad \rightsquigarrow \quad \dfrac{\dfrac{\vdash \Theta : A, F[t/x], \Gamma_1 \quad \vdash \Theta : B, \Gamma_2}{\vdash \Theta : A \otimes B, F[t/x], \Gamma_1, \Gamma_2}\,[\otimes]}{\vdash \Theta : A \otimes B, \exists x.F, \Gamma_1, \Gamma_2}\,[\exists]$$

- $(\exists/\oplus)$:

$$\dfrac{\dfrac{\vdash \Theta : A, F[t/x], \Gamma_1}{\vdash \Theta : A \oplus B, F[t/x], \Gamma}\,[\oplus_1]}{\vdash \Theta : A \oplus B, \exists x.F, \Gamma}\,[\exists] \quad \rightsquigarrow \quad \dfrac{\dfrac{\vdash \Theta : A, F[t/x], \Gamma}{\vdash \Theta : A, \exists x.F, \Gamma}\,[\exists]}{\vdash \Theta : A \oplus B, \exists x.F, \Gamma}\,[\oplus_1]$$

□

Definition 2.12 Let α be a rule with active formula F and G be a formula produced by α. Then we say that G is the *immediate descendant* of F and that all other formulas appearing in the premises of α are *immediate descendants* of the same formula appearing in α's conclusion. In a derivation, the transitive and reflexive closure of the *immediate descendant* relation specifies the *descendant* relation.

Definition 2.13 Let Ξ be a proof of a sequent $\mathcal{S}$. The *positive trunk* of Ξ is its largest derivation, with root $\mathcal{S}$, composed only of synchronous rules such that promotion rules produce leaves of the trunk. The border of a trunk is the set of its leaves, denoted as $\mathcal{B}(\Xi)$.

We distinguish each occurrence, F, created by a dereliction rule, by assigning to it a different number i, as in (F, i).

Definition 2.14 Given a positive trunk, Π, of the sequent $\vdash \Theta : \Gamma$, we assign to every occurrence of a dereliction rule, $D?$, in Π, a unique index (F, i) to the occurrence of formula F created. The active formulas in Π are the active formulas in Γ and the indexed formulas (F, i).

Definition 2.15 Let Π be a positive trunk of a proof with root sequent $\mathcal{S}$, let Δ be the set of all active formulas in Π, and let F and G be any two formulas in Δ. Then $F \prec_f G$ iff there exists a sequent in Π containing a negative descendant of F and a positive descendant of G.

The key observation of Saurin & Miller [Miller 2007b, Saurin 2008] is that whenever $F \prec_f G$ then one can safely focus on F before focusing on G. Hence, the main question is whether the relation $\prec_f$ has minimal elements or in other words, if the relation $\prec_f$ is acyclic. If so, then one can use the relation $\prec_f$ to determine which formulas to focus on first.

Lemma 2.16 *The relation $\prec_f$ is acyclic.*

Proof We prove this lemma by induction on the height of the positive trunk Π. Consider that $\mathcal{S}$ is the root sequent of Π. The base case is trivial since the asynchronous formulas in $\mathcal{S}$ are the minimal elements of $\prec_f$.

Inductive cases: We show that, after an application of any synchronous rule, the relation $\prec_f$ is still acyclic. The case for the rule 1 is easy since there are no premises. Hence, the relation $\prec_f$ is empty and trivially acyclic. For the synchronous rules with only one premise, *i.e.*, the rules $\oplus_i, \exists$ and !, we distinguish two cases: (i) if the immediate descendant of the active formula, F, is a minimal element, then it is easy to see that it must be the case that F is a minimal element of the relation $\prec_f$; (ii) otherwise, any minimal element of the premise continues to be a minimal element in the conclusion. For the synchronous rules that have more than one premise, *i.e.*, the $\otimes$ rule, we prove by contradiction: the application of this rule does not generate a cycle, since if it was the case that a cycle is generated, then it would have to be that the side-formulas and the subformulas of the principle formula, that are involved in the cycle, belong to the same branch, and hence it would be the case that in this branch $\prec_f$ also contains a cycle, which is a contradiction to the induction hypothesis.

It is easy to check that the dereliction rule does not cause any problem. Since all synchronous rules permute over dereliction rules, we can permute these rules eagerly in a positive trunk, until all dereliction rules, appearing in the trunk, are at the bottom of the trunk, as illustrates the following derivation, where Δ is a set indexed formulas:

$$\dfrac{\dfrac{\Pi'}{\vdash \Theta : \Gamma, \Delta}}{\vdots}$$
$$\dfrac{}{\vdash \Theta : \Gamma}\ [n \times D?]$$

The original positive trunk will have a cycle if and only if Π' contains a cycle, which is already ruled out from the discussion above. □

The corollary below follows immediately from the proof above.

Corollary 2.17 *If F is a minimal element in a positive trunk Π, then, in the premises of any synchronous rule, its subformulas are also minimal elements in their respective branches.*

Now that we formalized the focalisation graphs, we use permutation lemmas to transform any linear logic proof into a focused one, where a minimal element of the graph is focused on first.

Lemma 2.18 *Let Π be a positive trunk, $\mathcal{S}$ be the root sequent of Π such that that it cannot be the conclusion of any asynchronous rule, and F be a minimal element in Π. Then, if F is a formula created by a dereliction, then there is a proof of $\mathcal{S}$ where F is created and a rule is applied to it last. Otherwise, if F is not created by a dereliction, then there is a proof where a rule is applied to F last.*

Proof Since F is minimal, it must be the case that only synchronous rules separate the synchronous rule applied to F and the last rule. Furthermore, since we know that synchronous rules permute over

each other, we can permute the synchronous rule that is applied to F down so that it is applied last. If F is a formula created by a dereliction then we just permute first this rule down, and then the rule applied to F. □

Now we have all the pieces to complete the completeness proof. Given a linear logic proof, we check if it is possible to apply any asynchronous formula. If so, from the permutation lemmas, we can obtain a proof where this asynchronous rule is applied last. We repeat this process to its premises, until we are not able to apply any other asynchronous rule. At this point, from Lemma 2.18, we obtain a proof where a rule is applied to a minimal element in the resulting leaves. Repeat this process, until an asynchronous rule is applicable, and then start from the beginning of this procedure until there are no more open leaves. The resulting proof is a focused proof. □

Andreoli's completeness theorem states that, for *any assignment of polarities* to atoms, a formula F is provable in LLF if and only if it is provable in linear logic. Although the polarity assignment of literals does not affect provability, it does affect what synthetic connectives are available and, therefore, the shape and size of focused proofs. The polarity of literals affects the structure of proofs because the rules I_1 and I_2 explicitly refer to the polarity assigned to literals. Consider, for example, focusing on the positive formula $A^\perp \otimes N$ where formula N and atom A are both negative: this leads to the construction of two macro-rules for this synthetic connective

$$\dfrac{\dfrac{}{\vdash \Theta, A : \cdot \Downarrow A^\perp}\,[I_1] \quad \dfrac{\vdash \Theta, A : \Gamma \Uparrow N}{\vdash \Theta, A : \Gamma \Downarrow N}\,[R\Downarrow]}{\vdash \Theta, A : \Gamma \Downarrow A^\perp \otimes N}\,[\otimes] \qquad \dfrac{\dfrac{}{\vdash \Theta : A \Downarrow A^\perp}\,[I_2] \quad \dfrac{\vdash \Theta : \Gamma \Uparrow N}{\vdash \Theta : \Gamma \Downarrow N}\,[R\Downarrow]}{\vdash \Theta : \Gamma, A \Downarrow A^\perp \otimes N}\,[\otimes]$$

Thus, in order for focusing on the formula $A^\perp \otimes N$ to yield a successful derivation, it must be the case that the formula A is present in either the unbounded or bounded context. On the other hand, if the atom A is assigned positive polarity then the synthetic connective of $A^\perp \otimes N$ is introduced by a derivation of the form:

$$\dfrac{\dfrac{\vdash \Theta : \Gamma_1 \Uparrow A^\perp}{\vdash \Theta : \Gamma_1 \Downarrow A^\perp}\,[R\Downarrow] \quad \dfrac{\vdash \Theta : \Gamma_2 \Uparrow N}{\vdash \Theta : \Gamma_2 \Downarrow N}\,[R\Downarrow]}{\vdash \Theta : \Gamma_1, \Gamma_2 \Downarrow A^\perp \otimes N}\,[\otimes]$$

Here, there is no restriction imposed on A occurring in either the bounded or unbounded contexts.

One use of polarity assignment known in the literature is its relationship to forward and backward reasoning [Chaudhuri 2008b, Danos 1995, Dyckhoff 2007, Liang 2007] in intuitionistic logic. In particular, as illustrated in [Liang 2007], if all atoms are given negative polarity, the resulting proof system models backward chaining proof search and includes uniform proofs [Miller 1991]. If positive atoms are permitted as well, then forward chaining steps can also be accommodated. Consider, for example, the following two Horn clauses that specifies, in intuitionistic logic, the Fibonacci numbers:

$$\Delta = \{\mathit{fib}(0,0), \quad \mathit{fib}(1,1), \quad \forall n \forall x \forall y\, [\mathit{fib}(n,x) \wedge \mathit{fib}(n+1,y) \supset \mathit{fib}(n+2, x+y)]\}$$

where fib(n,N), denotes that N is the n^{th} Fibonacci number. Moreover, consider that we attempt to prove from Δ that the 12^{th} Fibonacci number is 144, denoted by fib(12,144). If we assign to all fib atoms negative polarity, then there is a unique focused proof, it is exponential in size and it has a *goal-directed/backchaining* behavior. On the other hand, if we assign to all fib atoms positive polarity, then there are infinitely many focused proofs, all of them have a *forward-chaining* behavior, and the smallest one is linear in size.

Although Andreoli considered only global polarity assignments, where all occurrences of a literal have the same polarity, Miller & Saurin noticed that the completeness proof also works with more flexible polarity assignments. They consider, for example, an occurrence based polarity assignment where only the occurrences of a literal that are in the same (sub)tree have necessarily the same

polarity. This implies that one can assign different polarities to different occurrences of the same literal. For example, in two different branches of a cut rule, with a literal cut formula, A, one can assign A as negative in the left branch and A as positive in the right branch:

$$\frac{\vdash \Theta : \Gamma_1 \Uparrow A \quad \vdash \Theta : \Gamma_2 \Uparrow A^{\perp}}{\vdash \Theta : \Gamma_1, \Gamma_2 \Uparrow \cdot} \; [Cut]$$

In Chapters 3 and 4 we will exploit the fact that by changing the polarity assignment for literals we obtain different *canonical* proofs.

Another important observation about focused proofs is that equivalent formulas can have completely different focusing behaviors. For example, one can show that the formulas A and $A \parr \perp$ are equivalent in linear logic. However, the latter formula is always negative, independent on the polarity of A. As adding $\parr \perp$ to formulas forces the polarity of a formula to be negative, we call it a *negative delay operator*. Similarly, $\otimes 1$ is a *positive delay operator*. Sometimes, one might be interested in stopping a focusing phase so that a synthetic connective does not become too *big*, that is, it is not introduced by many types of macro-rules. In these cases, it is useful to add such delay operators to subformulas. We use delay operators in Chapter 7.

We end this section by pointing out that there are many other systems, in different logics, that, as in LLF, provide a big-step reading of formulas. Later we will return to some such system, but for now, we invite a more interested reader to read [Liang 2008] and its references.

Chapter 3

Incorporating tables into proofs

We consider the problem of automating and checking the use of previously proved lemmas in the proof of some main theorem. In particular, we call the collection of such previously proved results a *table* and use a partial order on the table's entries to denote the (provability) dependency relationship between tabled items. Tables can be used in automated deduction to increase proof search efficiency, by storing previously proved subgoals, and in interactive theorem proving to store a sequence of lemmas introduced by a user to direct the proof system towards some final theorem. We incorporate tables into sequent calculus proofs by performing two steps. First, cuts are used to incorporate tabled items into a proof: one premise of the cut requires a proof of the lemma and the other premise has the lemma inserted into the set of assumptions. Second, to ensure that lemmas are not reproved, we exploit the non-canonicity of focusing systems and specify two specific *focusing disciplines* for tabled formulas.

1. We allow only atomic formulas in tables. Then, by exploiting the fact that, in *focusing systems*, atoms can be assigned positive or negative polarity, we impose the following polarity discipline: atoms that are in the table are given positive polarity and those not in the table are given negative polarity.

2. We restrict tabled formulas to fixed point literals, that is, fixed points and their negations, and universally quantified fixed point literals, denoting both finite successes and finite failures. Then, by exploiting the fact that fixed points can be *frozen* and be treated like atoms, we impose a second focusing discipline: fixed points that are instances of a tabled formula are frozen and those that are not instances of a tabled formula are not frozen.

We show that, in some fragments of first-order-logic, the only existing proofs and open derivations are those that do not attempt to (re)prove a tabled formula: for the first discipline, we use the *hereditary Harrop formulas* fragment [Miller 1987] of intuitionistic logic and for the second discipline, we use the fragment of intuitionistic logic with fixed points used in [Tiu 2005], where only formulas constructed from positive connectives appear in the left-hand-side of sequents and all fixed points are *noetherian*.

References: Parts of this chapter appeared in the conference paper [Miller 2007a] and in the informal proceedings [Nigam 2008a].

3.1 Introduction

A sequence of well chosen lemmas is often an important part of presenting a proof in, at least, informal mathematics. In some situations, one might feel that the sequence of lemmas itself could constitute an actual proof, particularly if the reader of the proof has significant mathematical means to fill in the gaps between the lemmas. Of course, as lemmas at the beginning of the list are proved, they can be used to help prove lemmas later in the list.

Although generating lemmas is a well known and critical activity in mathematical proof, most automated provers in the area of, say, logic programming, deductive databases, and model checking

usually search for proofs that do not contain lemmas: that is, when proofs are modeled using sequent calculus, these automated systems search for the so-called *cut-free* proofs. None-the-less, lemmas (and the corresponding *cut rule*) can play an important role in improving the search for or the presentation of proofs even in these kinds of systems.

Consider attempting to prove the conjunctive query $B \wedge C$ from a logic program Γ. This attempt can be reduced to first attempting to prove B from Γ and then C from Γ. It might well be the case that during the attempt to prove C, many subgoals need to be proved that were previously established during the attempt to prove B. Of course, if proved subgoals can be remembered from the first conjunct to the second, then it might be possible to build smaller proofs and these might be easier to find and to check for correctness. Some implemented logic programming systems introduce *tables* as a device to store proved literals so that their provability can be used in later attempts to prove goals. Systems such as XSB [Ramakrishna 1997] and Twelf [Pientka 2005] make it possible to specify that some predicates should be *tabled*: that is, whenever an atomic formula with such a predicate is successfully proved, that atomic formula is remembered by placing it in a global table. In this way, if the prover attempts to reprove an atom that is already tabled, then the proof process can be immediately stopped with a success. Besides this aspect of not proving formulas, tabling systems improve proof-search considerably by also remembering *finite failures*. Whenever a goal is shown not be provable, it is also stored in a table and if the interpreter attempts to prove a tabled finite failure, then it does not proceed.

In this chapter, we consider a general notion of *table* and attempt to show how proof theory can account for the following two salient aspects of tables.
(i) *Entering tabled formulas into the proof context.* Tables will be a partially ordered collections of formulas and proofs will be modeled using sequent calculus. The cut-rule will be used in a straightforward fashion to state the obligation to prove a tabled formula as well as insert that formula into the main proof context.
(ii) *Avoiding proving of tabled formulas.* It is easy to provide algorithmic means for making certain that formulas are not reproved or for not attempting to prove a finite failure (for example, prior to attempting a proof of a formula, check if that formula is in the table). More challenging is to find a purely proof theoretic solution in which the only proofs and open derivations that can be built are those that do not attempt to (re)prove formulas. We achieve this by imposing specific focusing disciplines. We distinguish two different cases. In the first case, we consider restricting tables to atomic formulas. Then, we exploit the fact that, in focusing systems for intuitionistic logic, atoms can be assigned positive or negative polarity and propose the following focusing discipline: tabled atoms are assigned positive polarity and the remaining atoms negative polarity. The idea will be that *positive atoms are always present in the context.* In the second case, we assume that tables contain only fixed point literals and universally quantified fixed point literals, denoting finite successes and finite failures. Then, we exploit the fact that fixed points can be *frozen*, that is, such fixed points cannot unfold, thus behaving like atoms, and propose the following focusing discipline: instances of tabled formulas are always frozen and the remaining fixed points are not frozen. Now, the intuition is that the *provability of a frozen fixed point is decided*, that is, it is already known if a fixed point is provable or not.

This chapter is structured as follows. Section 3.2 presents some examples that help to motivate particular connections between tables and proofs. Section 3.3 illustrates how tables can be inserted into proofs by using the multicut inference rule (a simple generalization of the cut rule). Section 3.4 presents the focused system LJF for intuitionistic logic that will be used as starting point for the construction of systems that can incorporate tables. In Section 3.5, we discuss different designs for focused proofs with cuts. In Section 3.6, we specify the first focusing discipline for tabled formulas, which is later used in Section 3.7 to ensure that tabled formulas are not reproved in some fragments of first-order-logic. In Section 3.8, we give a short introduction to a proof theoretic notion of fixed points. Section 3.9 presents the focused system LJF$^{\mu}$ that extends LJF with fixed points. In Section

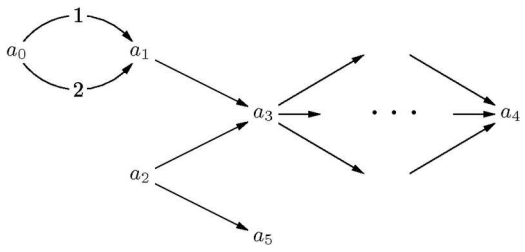


Figure 3.1: A directed graph: the ellipses represents a section of the graph with a large number of paths from a_3 to a_4.

3.10, we propose a focusing discipline for tabled fixed points which ensures that these formulas are not reproved. In Section 3.11, we speculate the use of tables as proof-objects. Finally in Sections 3.12 and 3.13, we illustrate the main results with some examples and conclude by pointing out some future works.

3.2 Some motivating examples

Consider the graph depicted in Figure 3.1 and assume that its arcs are represented by atomic facts of the form $(adj\ N_1\ N_2)$ where N_1 and N_2 are adjacent nodes in the graph. The following two Horn clauses can be used to describe when there is a path in this graph: $\forall x\,(path\,x\,x)$ and $\forall x \forall y \forall z\,(adj\ x\ z \wedge path\ z\ y \supset path\ x\ y)$.

Now consider attempting a proof of the conjunctive query $path\ a_1\ a_4 \wedge path\ a_2\ a_4$. The usual goal-directed logic interpreter will attempt to prove the two conjuncts independently. After making suitable backchaining steps, both independent attempts will give rise to the same subgoal $path\ a_3\ a_4$. The logic interpreter will then proceed to construct two (possibly identical) proofs of this subgoal. Clearly, a superior approach to proving this conjunctive goal would be to first prove the "lemma" $path\ a_3\ a_4$ and then make that lemma available to the proof of the original conjunctive goal.

A basic question here is: how does one ensure that the assumed lemma is not reproved? If there are special algorithmic connections between the logic interpreter and a tabling mechanism, as exist in, say, XSB [Ramakrishna 1997] and Twelf [Pientka 2005], then there are simple solutions to this problem of reproving lemmas. The question we are concerned with here, however, is whether or not there is an implementation independent and proof-theoretic solution to this problem of "reproof".

For a second example, consider the following possible approach to *memoization* that one could attempt to use in logic programming languages, such as λProlog, that contain implicational goals [Miller 1989]. Assume that the formula A is atomic and that we wish to prove the conjunction $A \wedge G$, for some general goal formula G. Since the attempt to prove G can reduce to several attempts to prove A, one might be tempted to rewrite the original conjunctive goal as the logically equivalent goal $A \wedge (A \supset G)$. In this rewritten goal, the assumption A is available during the attempt to prove G and, hence, if A appears as a subgoal to attempt, then the assumption is available to immediately close the proof. Unfortunately, when moving from $A \wedge G$ to $A \wedge (A \supset G)$, one is making proof search more non-deterministic since for every proof that proves A by matching with the assumed version of A, there is another proof where A is, in fact, reproved. As a result, this naive approach to memoization has never been successfully used in λProlog.

This example also allows us to notice that our concern for not reproving previously proved formulas is different from the concerns of relevance logic [Anderson 1975], a logic in which the nature of implication is changed so that hypotheses are *necessary* for the proof of conclusions. In the example above, if the attempt to prove G succeeds without using the assumption A, the implication $A \supset G$ is

still provable even if the assumption A is not "relevant" to the conclusion G.

Both of these examples illustrate a need for not only making proved formulas available for reuse but also enforcing that they are not reproved.

For a third example, consider again the graph depicted in Figure 3.1, but this time assume that we attempt to prove the goal $path\ a_2\ a_5$. By performing backchaining steps, an interpreter will attempt to prove either the subgoal $path\ a_3\ a_5$ or the subgoal $path\ a_5\ a_5$. While the latter subgoal is easily provable, the former subgoal is not provable and to check this fact, the interpreter would need to traverse all paths from a_3 to a_4. So depending on which backchaining step the interpreter performs, it would need more or less work to find a proof for this query. However, if an interpreter already had the knowledge (by looking in a table) that the subgoal $path\ a_3\ a_5$ is not provable, it could avoid proving it, increasing, hence, proof search efficiency. In this chapter, we also investigate proof-theoretic solutions to this problem of avoiding proving formulas known to be not provable.

3.3 Table as multicut derivation

In its most general form, a table is a partially ordered finite set of formulas.

Definition 3.1 A *table* is a tuple $\mathcal{T} = \langle \mathcal{A}, \preceq \rangle$, where $\mathcal{A}$ is some finite set of formulas and $\preceq$ is a partial order relation over the elements of $\mathcal{A}$.

A table is thus intended to be a structured collection of formulas, which are all provable from some fixed context. The relation $B \preceq C$ means that the formula B is provable and is used during a proof attempt of C: if the attempt to proof C leads to an attempt to prove B, that attempt can immediately stop successfully.

The following inference rule, called the *multicut* rule, is often used as a technical generalization to the cut rule to help prove cut-elimination theorems (see, for example, [Gentzen 1969, Slaney 1989]).

$$\frac{\Gamma \longrightarrow B_1 \quad \cdots \quad \Gamma \longrightarrow B_n \qquad B_1, \ldots, B_n, \Gamma \longrightarrow C}{\Gamma \longrightarrow C}\ [mc \quad (n \geq 0)]$$

Notice that if $n = 1$, this rule reduces to the usual cut-rule (for a single conclusion calculus), and if $n = 0$, this rule is essentially a simple "repetition." If $n \geq 1$, this rule can be seen as encoding n separate applications of the cut-rule. We say that the formulas $B_1, \ldots, B_n$ are the *cut-formulas* of this instance of this multicut rule.

The following definition describes how a table can be translated to a collection of multicut inference rules.

Definition 3.2 Let $\mathcal{T} = \langle \mathcal{A}, \preceq \rangle$ be a table. The *multicut derivation* for $\mathcal{T}$ and the sequent $\mathcal{S} = \Gamma \longrightarrow G$, written as $mcd(\mathcal{T}, \mathcal{S})$, is defined inductively as follows: if $\mathcal{A}$ is empty, then $mcd(\mathcal{T}, \mathcal{S})$ is the derivation containing just the sequent $\mathcal{S}$. Otherwise, if $\{A_1, \ldots, A_n\}$ is the collection of $\preceq$-minimal elements in $\mathcal{A}$ and if Π is the multicut derivation for the smaller table $\langle \mathcal{A} \setminus \{A_1, \ldots, A_n\}, \preceq \rangle$ and the sequent $\Gamma, A_1, \ldots, A_n \longrightarrow G$, then $mcd(\mathcal{T}, \mathcal{S})$ is the derivation

$$\frac{\Gamma \longrightarrow A_1 \quad \cdots \quad \Gamma \longrightarrow A_n \qquad \begin{matrix} \Pi \\ \Gamma, A_1, \ldots, A_n \longrightarrow G \end{matrix}}{\Gamma \longrightarrow G}\ [mc]$$

Multicut derivations are always *open* derivations: if the table $\mathcal{T}$ contains m elements, then a multicut derivation using $\mathcal{T}$ must have $m + 1$ open premises. A *proof* of a multicut derivation is any (closed) proof that extends that open derivation.

To illustrate this definition, consider the graph example in Section 3.2: let Γ contain the encoding of the original adjacency information as well as the specification of the *path* predicate, and consider the table that contains just the atomic formula $path\ a_3\ a_4$ and the trivial partial order on that formula. The following is the multicut derivation for that table and the sequent $\Gamma \longrightarrow path\ a_1\ a_4 \wedge path\ a_2\ a_4$:

$$\frac{\Gamma \longrightarrow path\ a_3\ a_4 \qquad \Gamma, path\ a_3\ a_4 \longrightarrow path\ a_1\ a_4 \wedge path\ a_2\ a_4}{\Gamma \longrightarrow path\ a_1\ a_4 \wedge path\ a_2\ a_4}\ [mc]$$

By using the cut, it was possible to introduce the lemma $path\ a_3\ a_4$ in the context of the rightmost branch. The left premise requires showing that there is, in fact, a path from a_3 to a_4, while the right branch attempts to show the original conjunctive goal under the assumption that this path exists. As we pointed out earlier, there are proofs of the right-most premise where this lemma is not reused but reproved. In the next sections, we exploit the notions of *focusing* and *polarity* in order to provide method for enforcing reuse.

In the rest of this chapter, we shall impose some restrictions to tables, namely, in Section 3.7, tables contain only atoms; and later, in Section 3.10, tables contain fixed points, negated fixed points, universally quantified fixed points, and universally quantified negated fixed points, where negation is intuitionistic, that is, $\neg F$ is defined as $F \supset \bot$.

3.4 Focusing and polarities via LJF

We present here the LJF focused proof system for intuitionistic logic of Liang & Miller [Liang 2007, Liang 2008]. The connectives of first-order intuitionistic logic will be the usual ones except that we shall use two conjunctions, written as $\wedge^+$ and $\wedge^-$: these two conjunctives are logically equivalent but their role in proof search will be different. (In particular, $\wedge^+$ resembles the linear logic multiplicative conjunction $\otimes$, while $\wedge^-$ resembles the additive conjunction &.) We shall classify the connectives $\wedge^-$, $\supset$, and $\forall$ as *asynchronous* (their right introduction rules are invertible), while the connectives $\wedge^+$, $\vee$, $\exists$, *true* and $\bot$ are classified as *synchronous* (their right introduction is not necessarily invertible).

As in linear logic, the synchronous/asynchronous dichotomy of non-atomic formulas will also be extended to atomic formulas: some atoms are to be considered as part of the asynchronous phase and the rest will be considered as synchronous. In the LJF proof system, a global and fixed assignment of positive or negative polarity to atomic formulas is assumed: such a mapping can give all atoms positive polarity or all atoms negative polarity, or, in fact, some atoms can be given positive polarity, while the others are given negative polarity.

The LJF focused proof system for intuitionistic logic is given by collecting together the inference rules in Figures 3.2 and 3.3. These figures use the following syntactic variables: A_n denotes a negative atom, A_p a positive atom, P a positive formula, N a negative formula, N_a an atom or a negative formula, and P_a an atom or a positive formula. Also Γ and Θ are both multisets of formulas and Γ contains only negative formulas and positive atoms. Finally, i is either 1 or 2 and y is not free in any formula of the conclusion of rules $\exists_l$ and $\forall_r$. The LJF proof system also has four types of sequents.

1. The sequent $[\Gamma], \Theta \longrightarrow \mathcal{R}$ is an *unfocused sequent*. Here, Γ contains negative formulas and positive atoms, and $\mathcal{R}$ is either a formula R or a bracketed formula $[R]$;
2. The sequent $[\Gamma] \longrightarrow [R]$ is an instance of the previous sequent where Θ is empty;
3. The sequent $[\Gamma] -_B\!\!\longrightarrow$ is a *right-focusing* sequent (the focus is B);
4. The sequent $[\Gamma] \xrightarrow{B} [R]$: is a *left-focusing* sequent (with focus on B).

As an inspection of the inference rules of LJF reveals, the search for an intuitionistic *focused* proof is composed of two alternating phases, as in linear logic. The *asynchronous phase* applies invertible

Asynchronous Phase

$$\frac{}{[\Gamma], \Theta, \bot \longrightarrow \mathcal{R}}\ [false_l] \qquad \frac{[\Gamma], \Theta \longrightarrow \mathcal{R}}{[\Gamma], \Theta, t \longrightarrow \mathcal{R}}\ [t_l] \qquad \frac{[\Gamma], \Theta, B, C \longrightarrow \mathcal{R}}{[\Gamma], \Theta, B \wedge^+ C \longrightarrow \mathcal{R}}\ [\wedge_l^+]$$

$$\frac{[\Gamma], \Theta \longrightarrow B \qquad [\Gamma], \Theta \longrightarrow C}{[\Gamma], \Theta \longrightarrow B \wedge^- C}\ [\wedge_r^-] \qquad \frac{[\Gamma], \Theta, B \longrightarrow \mathcal{R} \qquad [\Gamma], \Theta, C \longrightarrow \mathcal{R}}{[\Gamma], \Theta, B \vee C \longrightarrow \mathcal{R}}\ [\vee_l]$$

$$\frac{[\Gamma], \Theta, A \longrightarrow B}{[\Gamma], \Theta \longrightarrow A \supset B}\ [\supset_r] \qquad \frac{[\Gamma], \Theta, B \longrightarrow \mathcal{R}}{[\Gamma], \Theta, \exists y B \longrightarrow \mathcal{R}}\ [\exists_l] \qquad \frac{[\Gamma], \Theta \longrightarrow B}{[\Gamma], \Theta \longrightarrow \forall y B}\ [\forall_r]$$

Synchronous Phase

$$\frac{}{[\Gamma] -_t \rightarrow}\ [t_r] \qquad \frac{[\Gamma] -_{B_i} \rightarrow}{[\Gamma] -_{B_1 \vee B_2} \rightarrow}\ [\vee_r] \qquad \frac{[\Gamma] -_B \rightarrow \qquad [\Gamma] -_C \rightarrow}{[\Gamma] -_{B \wedge^+ C} \rightarrow}\ [\wedge_r^+]$$

$$\frac{[\Gamma] \xrightarrow{B_i} [R]}{[\Gamma] \xrightarrow{B_1 \wedge^- B_2} [R]}\ [\wedge_l^-] \qquad \frac{[\Gamma] -_A \rightarrow \qquad [\Gamma] \xrightarrow{B} [R]}{[\Gamma] \xrightarrow{A \supset B} [R]}\ [\supset_l]$$

$$\frac{[\Gamma] -_{B[t/x]} \rightarrow}{[\Gamma] -_{\exists x B} \rightarrow}\ [\exists_r] \qquad \frac{[\Gamma] \xrightarrow{B[t/x]} [R]}{[\Gamma] \xrightarrow{\forall x B} [R]}\ [\forall_l]$$

Figure 3.2: LJF rules for introducing logical connectives.

$$\frac{[N, \Gamma] \xrightarrow{N} [R]}{[N, \Gamma] \longrightarrow [R]}\ [D_l] \qquad \frac{[\Gamma] -_P \rightarrow}{[\Gamma] \longrightarrow [P]}\ [D_r] \qquad \frac{[\Gamma], P \longrightarrow [R]}{[\Gamma] \xrightarrow{P} [R]}\ [R_l] \qquad \frac{[\Gamma] \longrightarrow N}{[\Gamma] -_N \rightarrow}\ [R_r]$$

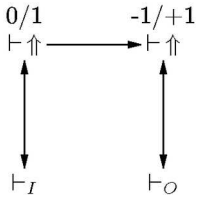
$$\frac{[\Gamma, N_a], \Theta \longrightarrow \mathcal{R}}{[\Gamma], \Theta, N_a \longrightarrow \mathcal{R}}\ [[]_l] \qquad \frac{[\Gamma], \Theta \longrightarrow [P_a]}{[\Gamma], \Theta \longrightarrow P_a}\ [[]_r] \qquad \frac{}{[\Gamma] \xrightarrow{A_n} [A_n]}\ [I_l] \qquad \frac{}{[\Gamma, A_p] -_{A_p} \rightarrow}\ [I_r]$$

Figure 3.3: The LJF structural rules.

(asynchronous) rules until exhaustion: no backtracking during this phase of search is needed. The asynchronous phase uses the first type of sequent above (the unfocused sequents): in that case, Θ may contain positive and negative formulas. The proof system for LJF processes the formulas in Θ as follows: if a given member of Θ is a negative formula or a positive atom, then that formula is moved to the Γ context (by using the $[]_l$ rule), also called the *bracketed* context of a sequent; otherwise, the formula is a positive non-atomic formula and an introduction rule (either $\wedge_l^+, \vee_l, \exists_l$ or $false_l$) is used to decompose it. The end of the asynchronous phase is represented by the second type of sequent. Such a sequent is then established by using one of the decide rules, D_r or D_l. The application of one of these decide rules then selects a formula for focusing and switches proof search to the *synchronous phase*, represented by the third and the fourth type of sequents. This phase then proceeds by applying sequences of inference rules on the unique focused formula: in general, backtracking may be necessary in this phase of search. Moreover, we classify the rule $\wedge_l^+, \wedge_r^-, \vee_l, \exists_l, false_l, \supset_r, \forall_r, [\cdot]_l, [\cdot]_r, R_l$ and R_r as *asynchronous rules* since they initiate the switch from focused to unfocused sequent (the R_l and R_r rules) or they propagate unfocused sequents upwards. The remaining rules are called *synchronous rules.*

Theorem 3.3 *Assume some arbitrary but fixed polarity assignment to atomic formulas. Let B be an intuitionistic logic formula and let B' be a formula that results from replacing different occurrences of $\wedge$ in B with either $\wedge^+$ or $\wedge^-$. The formula B is provable in intuitionistic logic if and only if B' is provable in LJF.*

Proof Here, we just sketch the proof given by Liang & Miller in [Liang 2008]. The soundness direction is straightforward, as it suffices to drop the focusing annotations. For the completeness direction, Liang & Miller use the *grand tour through linear logic*, illustrated by the following diagram:



To prove that a (focused) system, denoted by $\vdash_O$, is complete with respect to LJ, denoted by $\vdash_I$, one must provide two translations, 0/1 and -1/+1. The former translation is from unfocused intuitionistic logic formulas to focused linear logic. The latter translation is from the focused system (in this case LJF) to focused linear logic. Then, one finishes the proof by showing that any intuitionistic formula provable in linear logic when using the encoding 0/1 is also provable when using the encoding -1/+1. The translation 0/1 for LJ is depicted in Figure 3.4. A more interested reader can find also the -1/+1 translation for LJF in [Liang 2008]. □

3.5 Focused Proofs with Cuts

We now move our attention to focused proofs containing cuts. We distinguish two different types of cuts: *intraphase* cuts that can appear anywhere inside (*intra*) focusing phases and *interphase* cuts that can only appear between (*inter*) two focusing phases. The original LJF cuts, proposed by Liang & Miller [Liang 2008], are examples of intraphase cuts:

$$\frac{[\Gamma], \Theta \longrightarrow P \quad [\Gamma], \Theta', P \longrightarrow \mathcal{R}}{[\Gamma], \Theta, \Theta' \longrightarrow \mathcal{R}}\ [Cut^+] \qquad \frac{[\Gamma], \Theta \longrightarrow N_a \quad [\Gamma, N_a], \Theta' \longrightarrow \mathcal{R}}{[\Gamma], \Theta, \Theta' \longrightarrow \mathcal{R}}\ [Cut^-]$$

B	B^1	B^0	$(B^0)^\perp$
atom A	A	A	$A^\perp$
t	1	$\top$	0
$\perp$	0	0	$\top$
$P \wedge Q$	$!(P^1 \& Q^1)$	$!\,P^0 \& !\,Q^0$	$?(P^0)^\perp \oplus ?(Q^0)^\perp$
$P \vee Q$	$!\,P^1 \oplus !\,Q^1$	$!\,P^0 \oplus !\,Q^0$	$?(P^0)^\perp \& ?(Q^0)^\perp$
$P \supset Q$	$!(?(P^0)^\perp \parr Q^1)$	$!\,P^1 \multimap !\,Q^0$	$!\,P^1 \& ?(Q^0)^\perp$
$\exists x P$	$\exists x\, !\,P^1$	$\exists x\, !\,P^0$	$\forall x ?(P^0)^\perp$
$\forall x P$	$!\,\forall x P^1$	$\exists x\, !\,P^0$	$\exists x ?(P^0)^\perp$

Figure 3.4: The 0/1 translation used to encode LJ proofs into linear logic.

$$\frac{[\Gamma] \xrightarrow{B} [P] \quad [\Gamma], P \longrightarrow [R]}{[\Gamma] \xrightarrow{B} [R]}\,[Cut_1^{\leftarrow}] \qquad \frac{[\Gamma] \longrightarrow N \quad [\Gamma, N] \xrightarrow{B} [R]}{[\Gamma] \xrightarrow{B} [R]}\,[Cut_2^{\leftarrow}]$$

$$\frac{[\Gamma] -_{N_a}\!\!\rightarrow \quad [\Gamma, N_a] -_{R}\!\!\rightarrow}{[\Gamma] -_{R}\!\!\rightarrow}\,[Cut^{\rightarrow}]$$

The cut rules Cut^+ and Cut^- can appear in the middle of asynchronous phases, while the remaining cut rules can appear in the middle of synchronous phases. As Liang & Miller were mostly interested in showing that the cut-elimination theorem holds in LJF, they needed this collection of cuts as a technical device to construct the cut-elimination algorithm by permuting cuts upwards.

Although they are useful to formalize the cut-elimination procedure, intraphase cuts seem to have the wrong focusing flavor, from a proof search perspective. In focused systems with intraphase cuts, one can no longer perform a *big-step* reading of formulas, as one can introduce a *synthetic connective* (or a bipolar) with a derivation containing cuts, and, hence, the corresponding macro-rules would not only depend on the formulas in the conclusion sequent, but also on the cut-formulas introduced above by intraphase cuts.

On the other hand, *interphase* cut rules, appearing between focusing phases, do have a better focusing flavor, as one can still perform a big step reading of formulas.

$$\frac{[\Gamma] \longrightarrow [P] \quad [\Gamma], P \longrightarrow [R]}{[\Gamma] \longrightarrow [R]}\,[Cut_p] \qquad \frac{[\Gamma] \longrightarrow N \quad [\Gamma, N] \longrightarrow [R]}{[\Gamma] \longrightarrow [R]}\,[Cut_n]$$

These rules cannot be used inside a synchronous phase, since neither their premises nor their conclusions have a focused formula, nor can they be used inside an asynchronous phase, since their conclusion does not have any unbracketed formula. Moreover, one of their premises must be the conclusion of an asynchronous phase and the other premise must be either the conclusion of a synchronous phase or of another interphase cut rule. Hence, in general, any focused proof is now composed of three alternating phases: seeing from bottom-up, an asynchronous phase, where only asynchronous rules are applied, a *cut phase*, where only interphase cut rules are applied, and then two phases in parallel, a synchronous phase, where only synchronous rules are applied, and the following asynchronous phase.

Since tables are used to reduce the overall size of proofs, by eliminating redundant subproofs of the same goal, and we incorporate tables into proofs by using (multi)cut rules, we would like to use cut rules that do not limit how much we can "compress" proofs. We show that, in a fragment of logic, we can safely use interphase cuts, instead of intraphase cuts, and still obtain proofs of the same size. More precisely, for any given LJF proof that contains intraphase cuts, but no occurrences of the rules $\wedge_r^-$ and $\vee_l$, there is an LJF proof of the same sequent and of the same size that contains only interphase cuts.

Definition 3.4 Let Ξ be an LJF proof. The size of Ξ is the number of occurrences of logical and identity rules in Ξ.

Lemma 3.5 *All LJF synchronous rules permute over the cut rules $Cut^{\rightarrow}, Cut_1^{\leftarrow}$ and $Cut_2^{\leftarrow}$, without increasing the size of the proof.*

Proof We begin with the permutability cases with the $Cut^{\rightarrow}$ rule:

$$\dfrac{\dfrac{[\Gamma]\!-_{N_a}\!\!\rightarrow \quad [\Gamma, N_a]\!-_{A}\!\!\rightarrow}{[\Gamma]\!-_{A}\!\!\rightarrow}\,[Cut^{\rightarrow}] \qquad [\Gamma]\!-_{B}\!\!\rightarrow}{[\Gamma]\!-_{A\wedge^+ B}\!\!\rightarrow}\,[\wedge_r^+] \quad \rightsquigarrow \quad \dfrac{[\Gamma]\!-_{N_a}\!\!\rightarrow \qquad \dfrac{[\Gamma, N_a]\!-_{A}\!\!\rightarrow \quad [\Gamma, N_a]\!-_{B}\!\!\rightarrow}{[\Gamma, N_a]\!-_{A\wedge^+ B}\!\!\rightarrow}\,[\wedge_r^+]}{[\Gamma]\!-_{A\wedge^+ B}\!\!\rightarrow}\,[Cut^{\rightarrow}]$$

$$\dfrac{\dfrac{[\Gamma]\!-_{N_a}\!\!\rightarrow \quad [\Gamma, N_a]\!-_{A_i}\!\!\rightarrow}{[\Gamma]\!-_{A_i}\!\!\rightarrow}\,[Cut^{\rightarrow}]}{[\Gamma]\!-_{A_1\vee A_2}\!\!\rightarrow}\,[\vee_r] \quad \rightsquigarrow \quad \dfrac{[\Gamma]\!-_{N_a}\!\!\rightarrow \quad \dfrac{[\Gamma, N_a]\!-_{A_i}\!\!\rightarrow}{[\Gamma, N_a]\!-_{A_1\vee A_2}\!\!\rightarrow}\,[\vee_r]}{[\Gamma]\!-_{A_1\vee A_2}\!\!\rightarrow}\,[Cut^{\rightarrow}]$$

$$\dfrac{\dfrac{[\Gamma]\!-_{N_a}\!\!\rightarrow \quad [\Gamma, N_a]\!-_{A[t/x]}\!\!\rightarrow}{[\Gamma]\!-_{A[t/x]}\!\!\rightarrow}\,[Cut^{\rightarrow}]}{[\Gamma]\!-_{\exists x A}\!\!\rightarrow}\,[\exists_r] \quad \rightsquigarrow \quad \dfrac{[\Gamma]\!-_{N_a}\!\!\rightarrow \quad \dfrac{[\Gamma, N_a]\!-_{A[t/x]}\!\!\rightarrow}{[\Gamma, N_a]\!-_{\exists x A}\!\!\rightarrow}\,[\exists_r]}{[\Gamma]\!-_{\exists x A}\!\!\rightarrow}\,[Cut^{\rightarrow}]$$

Now we show the permutability cases with the $Cut_1^{\leftarrow}$ rule:

$$\dfrac{\dfrac{[\Gamma] \xrightarrow{A_i} [P] \quad [\Gamma], P \longrightarrow [R]}{[\Gamma] \xrightarrow{A_i} [R]}\,[Cut_1^{\leftarrow}]}{[\Gamma] \xrightarrow{A_1 \wedge^- A_2} [R]}\,[\wedge_l^-] \quad \rightsquigarrow \quad \dfrac{\dfrac{[\Gamma] \xrightarrow{A_i} [P]}{[\Gamma] \xrightarrow{A_1 \wedge^- A_2} [P]}\,[\wedge_l^-] \qquad [\Gamma], P \longrightarrow [R]}{[\Gamma] \xrightarrow{A_1 \wedge^- A_2} [R]}\,[Cut_1^{\leftarrow}]$$

$$\dfrac{\dfrac{[\Gamma] \xrightarrow{B} [P] \qquad [\Gamma], P \longrightarrow [R]}{[\Gamma] \xrightarrow{B} [R]}\,[Cut_1^{\leftarrow}] \qquad [\Gamma]\!-_{A}\!\!\rightarrow}{[\Gamma] \xrightarrow{A \supset B} [R]}\,[\supset_l] \quad \rightsquigarrow$$

$$\dfrac{[\Gamma] \xrightarrow{B} [P] \qquad [\Gamma]\!-_{A}\!\!\rightarrow}{[\Gamma] \xrightarrow{A \supset B} [P]}\,[\supset_l] \qquad [\Gamma], P \longrightarrow [R]}{[\Gamma] \xrightarrow{A \supset B} [R]}\,[Cut_1^{\leftarrow}]$$

$$\dfrac{\dfrac{[\Gamma] \xrightarrow{A[t/x]} [P] \qquad [\Gamma], P \longrightarrow [R]}{[\Gamma] \xrightarrow{A[t/x]} [R]}\,[Cut_1^{\leftarrow}]}{[\Gamma] \xrightarrow{\forall x A} [R]}\,[\forall_l] \quad \rightsquigarrow \quad \dfrac{\dfrac{[\Gamma] \xrightarrow{A[t/x]} [P]}{[\Gamma] \xrightarrow{\forall x A} [P]}\,[\forall_l] \qquad [\Gamma], P \longrightarrow [R]}{[\Gamma] \xrightarrow{\forall x A} [R]}\,[Cut_1^{\leftarrow}]$$

Finally the permutability cases with the $Cut_2^{\leftarrow}$ cut rule:

$$\dfrac{\dfrac{[\Gamma] \longrightarrow N \qquad [\Gamma, N] \xrightarrow{A_i} [R]}{[\Gamma] \xrightarrow{A_i} [R]}\,[Cut_2^{\leftarrow}]}{[\Gamma] \xrightarrow{A_1 \wedge^- A_2} [R]}\,[\wedge_l^-] \quad \rightsquigarrow \quad \dfrac{\dfrac{[\Gamma, N] \xrightarrow{A_i} [R]}{[\Gamma, N] \xrightarrow{A_1 \wedge^- A_2} [R]}\,[\wedge_l^-]}{[\Gamma] \xrightarrow{A_1 \wedge^- A_2} [R]}\,[Cut_2^{\leftarrow}]$$

$$\dfrac{[\Gamma]\!-_{A}\!\!\rightarrow \qquad \dfrac{[\Gamma] \longrightarrow N \qquad [\Gamma, N] \xrightarrow{B} [R]}{[\Gamma] \xrightarrow{B} [R]}\,[Cut_2^{\leftarrow}]}{[\Gamma] \xrightarrow{A \supset B} [R]}\,[\supset_l] \quad \rightsquigarrow \quad \dfrac{\dfrac{[\Gamma, N]\!-_{A}\!\!\rightarrow \quad [\Gamma, N] \xrightarrow{B} [R]}{[\Gamma, N] \xrightarrow{A \supset B} [R]}\,[\wedge_l^-]}{[\Gamma] \xrightarrow{A \supset B} [R]}\,[Cut_2^{\leftarrow}]$$

$$\dfrac{\dfrac{[\Gamma] \longrightarrow N \qquad [\Gamma, N] \xrightarrow{A[t/x]} [R]}{[\Gamma] \xrightarrow{A[t/x]} [R]}\,[Cut_2^{\leftarrow}]}{[\Gamma] \xrightarrow{\forall x A} [R]}\,[\forall_l] \quad \rightsquigarrow \quad \dfrac{\dfrac{[\Gamma, N] \xrightarrow{A[t/x]} [R]}{[\Gamma, N] \xrightarrow{\forall x A} [R]}\,[\forall_l]}{[\Gamma] \xrightarrow{\forall x A} [R]}\,[Cut_2^{\leftarrow}]$$

□

Lemma 3.6 *The cut rules Cut^+ and Cut^- permute, without increasing the size of the proof, over all LJF asynchronous rules except the rules $\wedge_r^-$ and $\vee_l$.*

Proof The permutation cases are similar to those used in the standard proof of cut-elimination. We show here some cases:

$$\dfrac{\dfrac{[\Gamma], A, B, \Theta \longrightarrow P}{[\Gamma], A \wedge^+ B, \Theta \longrightarrow P}\,[\wedge_l^+] \quad [\Gamma], \Theta', P \longrightarrow \mathcal{R}}{[\Gamma], A \wedge^+ B, \Theta, \Theta' \longrightarrow \mathcal{R}}\,[Cut^+] \quad \rightsquigarrow \quad \dfrac{[\Gamma], A, B, \Theta \longrightarrow P \quad [\Gamma], \Theta', P \longrightarrow \mathcal{R}}{\dfrac{[\Gamma], A, B, \Theta, \Theta' \longrightarrow \mathcal{R}}{[\Gamma], A \wedge^+ B, \Theta, \Theta' \longrightarrow \mathcal{R}}\,[\wedge_l^+]}\,[Cut^+]$$

$$\dfrac{\dfrac{[\Gamma], A, \Theta \longrightarrow N_a}{[\Gamma], \exists x.A, \Theta \longrightarrow N_a}\,[\exists_l] \quad [\Gamma, N_a], \Theta' \longrightarrow \mathcal{R}}{[\Gamma], \exists x.A, \Theta, \Theta' \longrightarrow \mathcal{R}}\,[Cut^-] \quad \rightsquigarrow \quad \dfrac{[\Gamma], A, \Theta \longrightarrow N_a \quad [\Gamma, N_a], \Theta' \longrightarrow \mathcal{R}}{\dfrac{[\Gamma], A, \Theta, \Theta' \longrightarrow \mathcal{R}}{[\Gamma], \exists x.A, \Theta, \Theta' \longrightarrow \mathcal{R}}\,[\exists_l]}\,[Cut^-]$$

$$\dfrac{[\Gamma], \Theta \longrightarrow N_a \quad \dfrac{[\Gamma, N_a], \Theta', A \longrightarrow B}{[\Gamma, N_a], \Theta' \longrightarrow A \supset B}\,[\supset_r]}{[\Gamma], \Theta, \Theta' \longrightarrow A \supset B}\,[Cut^-] \quad \rightsquigarrow \quad \dfrac{\dfrac{[\Gamma], \Theta, \longrightarrow N_a \quad [\Gamma, N_a], \Theta', A \longrightarrow B}{[\Gamma], \Theta, \Theta', A \longrightarrow B}\,[Cut^-]}{[\Gamma], \Theta, \Theta' \longrightarrow A \supset B}\,[\supset_r]$$

□

The rules $\vee_l$ and $\wedge_r^-$ cause a problem because of their additive behavior. So if we attempt to permute a cut over them, we need to copy proofs, as illustrates the following transformation:

$$\dfrac{\dfrac{[\Gamma], A, \Theta \longrightarrow N_a \quad [\Gamma], B, \Theta \longrightarrow N_a}{[\Gamma], A \vee B, \Theta \longrightarrow N_a}\,[\vee_l] \quad [\Gamma, N_a], \Theta' \longrightarrow \mathcal{R}}{[\Gamma], A \vee B, \Theta, \Theta' \longrightarrow \mathcal{R}}\,[Cut^-] \quad \rightsquigarrow$$

$$\dfrac{\dfrac{[\Gamma], A, \Theta, \longrightarrow N_a \quad [\Gamma, N_a], \Theta' \longrightarrow \mathcal{R}}{[\Gamma], A, \Theta, \Theta' \longrightarrow \mathcal{R}}\,[Cut^-] \quad \dfrac{[\Gamma], B, \Theta \longrightarrow N_a \quad [\Gamma, N_a], \Theta' \longrightarrow \mathcal{R}}{[\Gamma], B, \Theta, \Theta' \longrightarrow \mathcal{R}}\,[Cut^-]}{[\Gamma], A \vee B, \Theta, \Theta' \longrightarrow \mathcal{R}}\,[\vee_l]$$

In the derivation to the right, the proofs for the premises are duplicated increasing the size of the proof. Later, we restrict the language of logic programs so that $\vee_l$ and $\wedge_r^-$ rules are not used or restrict cuts to appear only at the bottom of proof trees.

Theorem 3.7 *Let Ξ be an LJF proof of a sequent S that contains intraphase cuts, but no occurrences of the rules $\wedge_r^-$ and $\vee_l$. Then, there is a proof of S of the same size that only contains interphase cuts.*

Proof We proceed by induction on the number of intraphase cuts. From the previous lemmas and the following transformations, we can construct from Ξ a proof of the same size as the original proof, but that contains only interphase cuts. We first permute all the cuts that appear inside a synchronous (respectively asynchronous) phase down (respectively up) to the beginning (respectively end) of the phase and then apply the transformations below, that replace instances of intraphase cuts by interphase cuts. Notice that these transformations do not increase the size of proofs.

$$\dfrac{[\Gamma] \xrightarrow{B} [P] \quad [\Gamma], P \longrightarrow [R]}{\dfrac{[\Gamma] \xrightarrow{B} [R]}{[\Gamma] \longrightarrow [R]}\,[D_l]}\,[Cut_1^{\leftarrow}] \quad \rightsquigarrow \quad \dfrac{\dfrac{[\Gamma] \xrightarrow{B} [P]}{[\Gamma] \longrightarrow [P]}\,[D_l] \quad [\Gamma], P \longrightarrow [R]}{[\Gamma] \longrightarrow [R]}\,[Cut_p]$$

$$\dfrac{[\Gamma] \longrightarrow N \quad [\Gamma, N] \xrightarrow{B} [R]}{\dfrac{[\Gamma] \xrightarrow{B} [R]}{[\Gamma] \longrightarrow [R]}\,[D_l]}\,[Cut_2^{\leftarrow}] \quad \rightsquigarrow \quad \dfrac{[\Gamma] \longrightarrow N \quad \dfrac{[\Gamma, N] \xrightarrow{B} [R]}{[\Gamma, N] \longrightarrow [R]}\,[D_l]}{[\Gamma] \longrightarrow [R]}\,[Cut_n]$$

$$\dfrac{\dfrac{\dfrac{[\Gamma] \longrightarrow N_a}{[\Gamma]{-_{N_a}\!\!\rightarrow}}\,[R_r] \quad [\Gamma, N_a]{-_R\!\!\rightarrow}}{[\Gamma]{-_R\!\!\rightarrow}}\,[Cut^{\rightarrow}]}{[\Gamma] \longrightarrow [R]}\,[D_r] \quad \rightsquigarrow \quad \dfrac{[\Gamma] \longrightarrow N_a \quad \dfrac{[\Gamma, N_a] \xrightarrow{B} [R]}{[\Gamma, N_a] \longrightarrow [R]}\,[D_l]}{[\Gamma] \longrightarrow [R]}\,[Cut_n]$$

$$\dfrac{\dfrac{\dfrac{}{[\Gamma]{-_{N_a}\!\!\rightarrow}}\,[I_r] \quad [\Gamma, N_a]{-_R\!\!\rightarrow}}{[\Gamma]{-_R\!\!\rightarrow}}\,[Cut^{\rightarrow}]}{[\Gamma] \longrightarrow [R]}\,[D_r] \quad \rightsquigarrow \quad \dfrac{\dfrac{}{[\Gamma] \longrightarrow N_a}\,[[]_r, D_r, I_r] \quad \dfrac{[\Gamma, N_a] \xrightarrow{B} [R]}{[\Gamma, N_a] \longrightarrow [R]}\,[D_l]}{[\Gamma] \longrightarrow [R]}\,[Cut_n]$$

$$\dfrac{\dfrac{[\Gamma] \longrightarrow [P]}{[\Gamma] \longrightarrow P}\,[[]_r] \quad [\Gamma], P \longrightarrow [R]}{[\Gamma] \longrightarrow [R]}\,[Cut^{+}] \quad \rightsquigarrow \quad \dfrac{[\Gamma] \longrightarrow [P] \quad [\Gamma], P \longrightarrow [R]}{[\Gamma] \longrightarrow [R]}\,[Cut_p]$$

$$\dfrac{[\Gamma] \longrightarrow N_a \quad [\Gamma, N_a] \longrightarrow [R]}{[\Gamma] \longrightarrow [R]}\,[Cut^{-}] \quad \rightsquigarrow \quad \dfrac{[\Gamma] \longrightarrow N_a \quad [\Gamma, N_a] \longrightarrow [R]}{[\Gamma] \longrightarrow [R]}\,[Cut_p]$$

□

3.6 A specific polarity discipline for tabling

3.6.1 How to provide an atom with a polarity?

We now return to the Fibonacci example discussed at the end of Chapter 2. Consider, for example, the Horn clause specification of the Fibonacci numbers f_n composed of the three formulas:

$$\Delta = \big\{ \mathit{fib}(0,0), \quad \mathit{fib}(1,1), \quad \forall n \forall x \forall y \left[\mathit{fib}(n,x) \wedge^{+} \mathit{fib}(n+1,y) \supset \mathit{fib}(n+2,x+y) \right] \big\}$$

If all atomic formulas are given negative polarity, then there exists only one focused proof of the sequent $[\Delta] \longrightarrow \mathit{fib}(n, f_n)$: that proof has size exponential in n and can be classified as a "backward chaining" proof. On the other hand, if all atomic formulas are given positive polarity, then there are an infinite number of focused proofs all of which are classified as "forward chaining" proofs: the smallest among these proofs is of size linear in n. Such exponential differences in size are clearly important when one is searching for proofs, as in, say, logic programming and automated reasoning. To see why there are bottom-up proofs of arbitrary size, let $j \geq 1$ and let $\mathcal{F}_j$ be the set of atomic formulas $\{\mathit{fib}(0,0), \mathit{fib}(1,1), \ldots, \mathit{fib}(j, f_j)\}$ and consider the following derivation within LJF.

$$\dfrac{\dfrac{\dfrac{}{[\Delta, \mathcal{F}_j]{-_{\mathit{fib}(i,g)}\!\!\rightarrow}}\,[I_r] \quad \dfrac{}{[\Delta, \mathcal{F}_j]{-_{\mathit{fib}(i+1,h)}\!\!\rightarrow}}\,[I_r] \quad \dfrac{[\Delta, \mathcal{F}_j, \mathit{fib}(i+2, g+h)] \longrightarrow \mathit{fib}(100, f_{100})}{[\Delta, \mathcal{F}_j] \xrightarrow{\mathit{fib}(i+2,g+h)} \mathit{fib}(100, f_{100})}\,[R_l]}{[\Delta, \mathcal{F}_j] \xrightarrow{\forall n \forall x \forall y \left[\mathit{fib}(n,x) \wedge^{+} \mathit{fib}(n+1,y) \supset \mathit{fib}(n+2,x+y)\right]} \mathit{fib}(100, f_{100})}\,[\forall_l, \supset_l]}{[\Delta, \mathcal{F}_j] \longrightarrow \mathit{fib}(100, f_{100})}\,[D_l]$$

In this derivation, the decide rule selected the clause providing the recursive calls of the logic program. After instantiating the binders n, x, and y with integers i, g, and h, we are forced (by polarity considerations) to declare that $\mathit{fib}(i,g)$ and $\mathit{fib}(i+1,h)$ are members of $\mathcal{F}_j$. Thus, $0 \leq i \leq j-1$. Furthermore, proof search continues by proceeding up the right-most branch of the proof: thus, the effect of focusing on the clause for recursion is that the sequent $[\Delta, \mathcal{F}_j] \longrightarrow \mathit{fib}(100, f_{100})$ is reduced to the sequent $[\Delta, \mathcal{F}_j, \mathit{fib}(i+2, g+h)] \longrightarrow \mathit{fib}(100, f_{100})$. If $i = j-1$ then this latter sequent is equal to $[\Delta, \mathcal{F}_{j+1}] \longrightarrow \mathit{fib}(100, f_{100})$ and in this case, we have made process in computing more of the table of Fibonacci numbers. However, in the case that $0 \leq i \leq j-2$, then $\mathcal{F}_j \cup \{\mathit{fib}(i+2, g+h)\} = \mathcal{F}_j$, in which case, no progress has been made in that computational effort. Of course, such non-progressing inference steps can be repeated arbitrarily. (We return to this issue in Section 3.6.3.).

An example of mixing polarity assignments was given by Jagadeesan, Nadathur, and Saraswat [Jagadeesan 2005]. They describe a proof system that allows one to dictate the use of backward and forward chaining in different parts of a proof system. Backward chaining was used to model proof search in logic programming (particularly, in λProlog) and forward chaining was used to model constraint based reasoning (as in concurrent constraint programming). Their proof system can be embedded and generalized within LJF [Liang 2007] by assigning to the atoms that represent constraints a positive polarity and to the other atoms that represent the rest of the logic program a negative polarity.

In this chapter, we consider another approach to polarity assignment that can be motivated operationally as follows: a typical logic programming interpreter has, at any moment, a number of goals to attempt conjunctively. If the interpreter succeeds in proving one of these goals, that goal can be "tabled"; that is, remembered as a success that can be used to help prove other goals. We shall link polarity of atoms with membership in the table: an atom has negative polarity if it is not in the table (the default assumption when the search for a proof starts) and has positive polarity once it is proved and added to the table. Another aspect of this operational description is that the polarity of some atoms can *change* from negative to positive. The examples mentioned before were based on a global and rigid polarity assignment.

3.6.2 The LJF^t proof system

We now need to translate the operational intuition for polarity assignment given above to a more declarative and proof theoretic one. We do this by building the proof system LJF^t from the system LJF that attempts to incorporate a table.

Since polarity assignment is not fixed, we will have every sequent carry its own polarity assignment. In particular, we add to the sequents of LJF a set, written usually with the syntactic variable $\mathcal{P}$, that contains the atoms that have been declared to have a positive polarity. The three kinds of sequents in LJF^t are given as follows:

$$\mathcal{P}; [\Gamma], \Theta \longrightarrow \mathcal{R} \qquad \mathcal{P}; [\Gamma] -_B\!\!\rightarrow \qquad \mathcal{P}; [\Gamma] \xrightarrow{B} [R]$$

An occurrence of an atom within a given sequent is declared as positive if it occurs in $\mathcal{P}$ and is declared as negative otherwise.

The inference rules of LJF are modified as followed to yield the LJF^t proof system: in each case, the inference rule of the LJF^t proof system takes the same name from the corresponding rule in LJF.

1. The introduction rules in Figure 3.2 and the bracket rules $[]_r$ and $[]_l$ translate directly to the corresponding inference rules in LJF^t by simply adding the "$\mathcal{P}$;" prefix to all sequents in the rule.

2. The decide rules D_l, D_r, the release rules R_l, R_r, and the initial rules I_l and I_r in Figure 3.2 are similarly modified by the addition of the "$\mathcal{P}$;" prefix. In these cases, however, the notion of negative and positive formula may require checking if an atomic formula is or is not a member of the $\mathcal{P}$ declaration. For example, the two initial rules are given explicitly as the following two inferences.

$$\frac{}{\mathcal{P}; [\Gamma] \xrightarrow{A} [A]}\ [I_l,\ \text{provided } A \notin \mathcal{P}] \qquad \frac{}{\mathcal{P}; [A, \Gamma] -_A\!\!\rightarrow}\ [I_r,\ \text{provided } A \in \mathcal{P}]$$

Besides these rather obvious and direct embellishments of sequents and inference rules using this new declaration, we add one more inference rule. In particular, we add a "polarized" version of the *atomic* interphase multicut rule *mc* given below, where $A_1, \ldots, A_n$ are atomic formulas:

$$\frac{\mathcal{P}; [\Gamma] \longrightarrow A_1 \quad \cdots \quad \mathcal{P}; [\Gamma] \longrightarrow A_n \qquad \mathcal{P} \cup \{A_1, \ldots, A_n\}; [\Gamma, A_1, \ldots, A_n] \longrightarrow [R]}{\mathcal{P}; [\Gamma] \longrightarrow [R]}\ [mc,]$$

In that inference rule, the atomic formulas $A_1, \ldots, A_n$ are proved in the left-premises from the same set of assumptions Γ, containing the same table (*i.e.*, positive polarity assignment $\mathcal{P}$). If we read that inference rule bottom-up, then the search for a proof of the formula R moves from one with table $\mathcal{P}$ to one with table $\mathcal{P} \cup \{A_1, \ldots, A_n\}$. Notice that if a proof has no occurrences of the multicut rule then the context "$\mathcal{P};$" is the same in all sequent occurrences in that proof.

Lemma 3.8 *Let Ξ be a cut-free LJF^t proof of the sequent $\mathcal{S}$, whose polarity context is $\mathcal{P}$, and let $\mathcal{S}'$ be an arbitrary sequent in Ξ. Then, the polarity context of $\mathcal{S}'$ is equal to $\mathcal{P}$.*

Proof Simple induction on the height of cut-free proofs. □

As a rule, we always consider that the polarity context of the endsequent of an LJF^t proof is empty, that is, its table is empty. To incorporate a table into a LJF^t proof, we use the multicut derivation obtained from it as described in Section 3.3. Only that we now use the polarized multicut rule above to construct these derivations. Given these constraints, it is easy to check that it is always the case that in every sequent of a proof the polarity context is contained in the bracketed context.

Lemma 3.9 *Let Ξ be an LJF^t proof of the sequent $\emptyset; [\cdot] \longrightarrow F$ and let $\mathcal{S}$ be an arbitrary sequent in Ξ with polarity context $\mathcal{P}$ and bracketed context Γ. Then, $\mathcal{P} \subseteq \Gamma$.*

Proof Simple induction on the height of proofs. The polarity context can only change when the polarized multicut rule *mc* is used. □

Proposition 3.10 *Let B be some intuitionistic formula and let B' be a formula that results from replacing different occurrences of $\wedge$ in B with either $\wedge^+$ or $\wedge^-$. Then, the set of atoms $\mathcal{P}$ intuitionistically entails the formula B if and only if the sequent $\mathcal{P}; [\mathcal{P}] \longrightarrow B'$ is provable in LJF^t.*

Proof The soundness direction is straightforward. We just need to remove all focusing annotations. For the completeness direction we use the completeness result for LJF (Theorem 3.3): since the polarity assignment of atoms in LJF does not affect provability and LJF is complete and admits cut elimination, there is a cut-free LJF proof of the sequent $[\mathcal{P}] \longrightarrow B'$ where all atoms in $\mathcal{P}$ are assigned with positive polarity and the remaining atoms with negative polarity. Now, we just add the polarity context $\mathcal{P}$ to all sequents in this LJF proof and obtain a proof of $\mathcal{P}; [\mathcal{P}] \longrightarrow B'$ in LJF^t. □

3.6.3 Dropping the release-left rule

There is, however, a serious problem remaining to be addressed regarding "reproving": recall again the Fibonacci example of Subsection 3.6.1. When doing forward chaining from a database, any atom that is inferred via forward chaining can be continually reproved and added again and again to the left-hand-side context. We now address this source of reproving.

Given the polarity discipline maintained in LJF^t proofs, there is a connection between the notion of polarity assignment, which involves (prior) provability of an atomic formula, and general (focused) provability, which uses polarity assignments to decide the shape of proofs. This linkage provides us with an opportunity to make an optimization to the LJF^t proof system. Consider an instance of the "release-left" rule, namely,

$$\dfrac{\dfrac{\mathcal{P}; [\Gamma, P] \longrightarrow [R]}{\mathcal{P}; [\Gamma], P \longrightarrow [R]}\,[[]_l]}{\mathcal{P}; [\Gamma] \xrightarrow{P} [R]}\,[R_l]$$

where P is a positive polarity atomic formula. The assumption that P is a positive atom means that $P \in \mathcal{P}$ and combining this with the invariant $\mathcal{P} \subseteq \Gamma$ means that $\Gamma \cup \{P\} = \Gamma$. Thus, whenever the left-focus is on a positive atom, we can be guaranteed that we have made no progress in proof search, since that atom is already part of the set of hypotheses. Hence, to disallow reproving a positive

polarity atom via forward-chaining steps, we ensure that proof search in this case fails: one way to do that is to remove all occurrences of the release-left rule R_l in which the focus is on a positive atom. In particular, let LJF^t_- proof system to be the result of deleting those occurrences of the release-left rule R_l from LJF^t.

Notice that one could also imagine disallowing $[]_l$ to bracket positive polarity atoms. However, this restriction would be too strong, as it would disallow not only forward chaining proofs, but also desired proofs, such as when a goal has a positive polarity atom as an *hypothetical assumption*. For example, the sequent $\{A\}; [A] \longrightarrow A \supset G$ would not be provable if we restrict the rule $[]_l$ in the way described before.

In order to prove the completeness of LJF^t_- (Proposition 3.12), we need the following lemma, which is proved by a simple induction on the structure of LJF^t proofs.

Lemma 3.11 *The border of the positive trunk of an LJF^t-proof of $\mathcal{P}; [\Gamma] \xrightarrow{F} [G]$ contains at most one left-focusing sequent and that sequent is of the form $\mathcal{P}; [\Gamma] \xrightarrow{B} [G]$, for some formula B. The border of the positive trunk of an LJF^t-proof of a right-focusing sequent contains no left-focusing sequent.*

Proposition 3.12 *Let B be some intuitionistic formula and let B' be a formula that results from replacing different occurrences of $\wedge$ in B with either $\wedge^+$ or $\wedge^-$. Then, the set of atoms $\mathcal{P}$ intuitionistically entails B if and only if there is an LJF^t_- proof of $\mathcal{P}; [\mathcal{P}] \longrightarrow B'$.*

Proof Soundness is again straightforward. We just need to remove all focusing annotations. For completeness, assume that $\mathcal{P}$ is a finite set of atoms and that they intuitionistically entail B. Then, by the completeness of LJF^t proof (Proposition 3.10), there is an LJF^t proof Ξ of $\mathcal{P}; [\mathcal{P}] \longrightarrow B'$. We now proceed by induction on the number of occurrences in Ξ of R_l inference rules applied to positive atoms. If that number is 0, then Ξ is also an LJF^t_- proof. Otherwise, an occurrence of the R_l inference rule on a positive atomic formula A must occur on the border of the positive trunk of a left-focused sequent of a sequent, say $\mathcal{P}; [\Gamma] \longrightarrow [G]$, in Ξ. Such an occurrence has the following shape, modulo the order of the premises.

$$\dfrac{\dfrac{\dfrac{\Xi_0}{\mathcal{P}; [\Gamma, A] \longrightarrow [G]}\,[R_l, []_l]}{\mathcal{P}; [\Gamma] \xrightarrow{A} [G]} \qquad \dfrac{\Xi_1}{\mathcal{P}; [\Gamma] -_{G_1}\!\!\rightarrow} \quad \cdots \quad \dfrac{\Xi_n}{\mathcal{P}; [\Gamma] -_{G_n}\!\!\rightarrow}\,[\supset_l, m \times \wedge^+_r]}{\dfrac{\mathcal{P}; [\Gamma] \xrightarrow{F} [G]}{\mathcal{P}; [\Gamma] \longrightarrow [G]}\,[D_l]}$$

Here, $n \geq 0$. Given that $\mathcal{P} \subseteq \Gamma$, it is the case that Ξ_0 is also a proof of the root sequent $\mathcal{P}; [\Gamma] \longrightarrow [G]$: hence, the entire proof figure above can be replaced by Ξ_0. As a result, the number of occurrences of the R_l rule applied to positive atoms has been reduced by at least one. □

3.7 Tables of finite successes

3.7.1 The Horn clause case

Consider the following description of two classes of (annotated) formulas.

$$G := A \mid \mathit{true} \mid G_1 \wedge^+ G_2 \mid G_1 \vee G_2 \mid \exists x\, G$$
$$D := A \mid D_1 \wedge^- D_2 \mid G \supset D \mid \forall x\, D$$

Here, A-formulas are atomic formulas, D-formulas are Horn clauses, and G-formulas are the *goals* or *queries* that are used as the body of the Horn clauses. Notice that the G-formulas allow arbitrary

occurrences of right-synchronous connectives (that is, a connective whose right-introduction rules is a right-focus rule in Figure 3.2). Similarly, the top-level connectives of D-formulas are the left-synchronous connectives (that is, a connective whose left-introduction rules is a left-focus rule in Figure 3.2). The G-formulas are encoded using synchronous connectives only, while the D formulas are encoded using asynchronous connectives only: since a G-formula can occur negatively within a D formula, the synchronous connectives of the embedded G-formula add to D-formulas asynchronously. Therefore, LJF^t_- positive trunks concluding a sequent which focuses on a D-formula on the left have to be of the following form, modulo the order of the premises:

$$\frac{\mathcal{P}; [\Gamma] \xrightarrow{A} [R] \qquad \mathcal{P}; [\Gamma] -_{A_1} \rightarrow \qquad \cdots \qquad \mathcal{P}; [\Gamma] -_{A_n} \rightarrow}{\mathcal{P}; [\Gamma] \xrightarrow{D} [R]} [\supset_l, m \times \wedge_r^+]$$

Here, $n \geq 0$ and $A, A_1, \ldots, A_n$ are atomic formulas. Moreover, when it comes to sequents $[\Gamma] \longrightarrow B$ that we shall consider in this Horn clause case, the formula B will always be a goal formula and Γ will always be a set of Horn clauses. As a result, proofs involving Horn clauses will have asynchronous phases that are essentially empty.

Proposition 3.13 *Let $\mathcal{P}$ be a set of atomic formulas, let A be an atom, and let Γ be a finite set of Horn clauses. Let Ξ be any LJF^t_- proof of the sequent $\mathcal{P}; [\Gamma] \longrightarrow A$. Then every sequent occurring in Ξ is either of the form*

$$\mathcal{P}'; [\mathcal{P}', \Gamma] \longrightarrow A', \qquad \mathcal{P}'; [\mathcal{P}', \Gamma] \longrightarrow [A'], \qquad \mathcal{P}'; [\mathcal{P}', \Gamma] -_G \rightarrow, \qquad \text{or} \qquad \mathcal{P}'; [\mathcal{P}', \Gamma] \xrightarrow{D} A',$$

where A' is some atomic formula and $\mathcal{P}'$ is the disjoint union of $\mathcal{P}$ and some set $\mathcal{P}''$ of atomic formulas.

This proposition is proved by a simple induction over the structure of LJF^t_- proofs. It is interesting to notice the following about restricting our attention to Horn clauses.

1. There are no occurrences of asynchronous inference rules in Ξ: that is, if Ξ contains the sequent $\mathcal{P}'; [\Gamma'], \Delta \longrightarrow B$ then Δ is empty and B is atomic.

2. Finite sets of Horn clauses Γ can be restricted to just a single Horn clause, namely $H = \bigwedge^-_{D \in \Gamma} D$. If this is done, the decide-left rule D_l, when used to prove the sequent $\mathcal{P}'; [\mathcal{P}', \Gamma] \longrightarrow [A']$, picks between an atom in the "table" $\mathcal{P}'$ and the (entire) logic program H. The $\wedge^-_l$ is then used to select which member of Γ to (continue to) use as its focus.

3. When moving from conclusion to premise, if the left-hand context changes, then the inference rules where that change occurs is an instance of the *mc* rule. In cut-free proofs, the left-hand sides of all sequents occurrences are fixed.

We now show that when an atom is tabled, that is, it is assigned positive polarity and is in the bracketed context of the endsequent, then the only possible proofs are those for which any subproof of this tabled atom is bounded by a small number of decide rules.

Definition 3.14 The *decide-depth* of a focused proof Ξ is the maximum number of occurrences of decide rules (*i.e.*, D_r and D_l) on any path from the root to a leaf in Ξ.

Proposition 3.15 *Let Γ be a finite set of Horn clauses, G be a G-formula, $\mathcal{P}$ be a set of atoms, and $A \in \mathcal{P}$ be an atom. Let Ξ be an arbitrary LJF^t_- cut-free proof of the sequent $\mathcal{P}; [\mathcal{P}, \Gamma] \longrightarrow G$. Then all occurrences of the sequent $\mathcal{P}; [\mathcal{P}, \Gamma] \longrightarrow A$ and $\mathcal{P}; [\mathcal{P}, \Gamma] -_A \rightarrow$, in Ξ, are the conclusion of a derivation of decide-depth of at most one.*

Proof We check all possible ways one can prove the sequent $\mathcal{P}; [\mathcal{P}, \Gamma] \longrightarrow A$. The other case, for the sequent $\mathcal{P}; [\mathcal{P}, \Gamma] -_A \rightarrow$, is similar.

Seen bottom-up, the first rule must be $[]_r$ since the atom A is positive. Now we check what are the possible ways to conclude the sequent $\mathcal{P}; [\mathcal{P}, \Gamma] \longrightarrow [A]$. There are two alternatives, either focus on the left on a D-formula, or on the right. Let us consider the former case first. As described above, if we focus on the left on a D-formula, this formula is completely decomposed, yielding a premise where an atom is focused on the left. However, it cannot be the case that this premise is the conclusion of any rule in LJF^t_-: it cannot be the conclusion of the initial left rule because the atom A is on the right-hand-side and has positive polarity; nor can it be the conclusion of the R_l rule because R_l is restricted to non-atomic formulas. The only remaining option is to focus, instead, on the right, which can then only be the conclusion of the initial rule I_r. □

The proof theory that we have been presenting here deals directly only with the *use* of tables within proofs and not with their *discovery*. We now consider briefly an example of how a table can be built in the case of Horn clauses. Assume that all atoms are given a negative polarity and let Ξ be a cut-free LJF proof of $[\Gamma] \longrightarrow [G]$, where Γ is a finite set of Horn clauses and G is some G-formula. Notice that if Ξ contains the unfocused sequent $[\Gamma'] \longrightarrow [G']$ for G-formula G', then $\Gamma' = \Gamma$. Let table $\mathcal{T}$ be the ordered set of atoms $\langle \mathcal{A}, \preceq \rangle$ where $\mathcal{A}$ is the set of atomic formulas A such that $[\Gamma] \longrightarrow [A]$ occurs in Ξ and where $\preceq$ is defined as follows: let $\preceq'$ be the order relation defined on *occurrences* of atoms that is given by the post-order traversal (*i.e.*, process a node's premises before processing the node) of Ξ. Then, $A \preceq A'$ is defined to hold if and only if there is an occurrence of atom A that is $\preceq'$-related to all occurrences of atom A'.

The following proposition shows that it is trivial to extend a multicut derivation that is built from such a table.

Proposition 3.16 *Let Γ be a set of Horn clauses and let Ξ be a LJF cut-free proof of $[\Gamma] \longrightarrow [G]$, where all atoms have negative polarity. Let $\mathcal{T}$ be a table obtained from Ξ using the post-order traversal described above. There exists a proof for $mcd(\mathcal{T}, [.]; \Gamma \longrightarrow G)$ such that all of its added subproofs have decide-depth of at most one.*

Proof Proof by induction on the length of the table's longest chain. In the base case, such a chain contains just one atomic formula, yielding a multicut $\mathcal{M}$ containing only one cut. To complete the open premises of $\mathcal{M}$, one would need to decide on an atom that is already present in the context of the proof, and, therefore, completing $\mathcal{M}$ with derivations containing only one decide rule.

Now the inductive step: Consider that the sequent $\mathcal{P}; [\Gamma \cup \mathcal{P}] \longrightarrow [A_i]$ is a sequent branch in the multicut derivation $\mathcal{M}$. We can find a proof for this sequent, with one decide left rule, by proceeding as follows: we check in Ξ the formula F that was focused on to prove A_i. After this rule is performed, it has to be the case that the body of F is decomposed and finishes with previously proved atoms, that is, a positive atom, focused on the right, and, therefore, the proof must finish with initial right rules. □

We can extend, in a straightforward way, the table extraction algorithm, described before, to also extract tables from LJF proofs that have an *arbitrary* polarity assignment. The key observation is that we can simulate forward chaining steps in an LJF proof by using multicuts in the corresponding LJF^t_- proof, as illustrates the following transformation, where the atom A has positive polarity on the derivation on the left and $\mathcal{P}' = \mathcal{P} \cup \{A\}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Xi}{[\Gamma, A] \longrightarrow [G']}\,[R_r, []_l]}{[\Gamma] -_G \rightarrow \quad [\Gamma] \xrightarrow{A} [G']}\,[\supset_l]}{[\Gamma] \xrightarrow{G \supset A} [G']}\,[D_l]}{[\Gamma] \longrightarrow [G']} \quad \rightsquigarrow \quad \dfrac{\dfrac{\mathcal{P}; [\Gamma] -_G \rightarrow \quad \dfrac{}{\mathcal{P}; [\Gamma] \xrightarrow{A} [A]}\,[I_l]}{\mathcal{P}; [\Gamma] \xrightarrow{G \supset A} [A]}\,[\supset_l]}{\mathcal{P}; [\Gamma] \longrightarrow A}\,[[]_r, D_l] \qquad \mathcal{P}'; [\Gamma, A] \longrightarrow [G']}{\mathcal{P}; [\Gamma] \longrightarrow [G']}\,[mc]$$

We modify the algorithm above by also checking in the LJF proof for forward chaining steps, that is, when R_l rule is applied to a positive atom A. In this case, to construct the partial ordering $\preceq'$, we process A before we process the nodes in Ξ.

As an example of building such a table, consider the Fibonacci example from Section 3.6.1. By assigning all atomic formulas negative polarity, a cut-free proof of $fib(n, f_n)$ has size exponential in n. The table constructed from that proof, however, has size linear in n: in fact, the table encodes the list

$$fib(0,0), fib(1,1), \ldots, fib(n, f_n).$$

The multicut-derivation built from that table has $n+1$ subproofs, all of which are essentially trivial proofs that validates that, for example, $fib(i+2, F_{i+2})$ follows from $fib(i+1, F_{i+1})$ and $fib(i, F_i)$ by a simple addition.

To further illustrate, consider the example in Section 3.2, where the subgoal $path\ a_3\ a_4$ is tabled: that is, $path\ a_3\ a_4 \in \mathcal{P}$. Any proof of the rightmost branch of the multicut derivation obtained will never reprove the lemma $path\ a_3\ a_4$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathcal{P}; [\Gamma, path\ a_3\ a_4] \xrightarrow{adj\, a_1\, a_3} [adj\, a_1\, a_3]}\,[I_l]}{\mathcal{P}; [\Gamma, path\ a_3\ a_4] \longrightarrow [adj\, a_1\, a_3]}\,[D_l]}{\mathcal{P}; [\Gamma, path\ a_3\ a_4] -_{adj\, a_1\, a_3}\!\!\rightarrow}\,[R_r] \qquad \dfrac{}{\mathcal{P}; [\Gamma, path\ a_3\ a_4] -_{path\ a_3\ a_4}\!\!\rightarrow}\,[I_r]}{\dfrac{\mathcal{P}; [\Gamma, path\ a_3\ a_4] -_{adj\, a_1\, a_3 \wedge^+ path\ a_3\ a_4}\!\!\rightarrow}{\mathcal{P}; [\Gamma, path\ a_3\ a_4] \longrightarrow [path\ a_1\ a_4]}}\,[\wedge_r^+]$$

The memoization example of Section 3.2 can be addressed similarly: instead of doing the goal reduction illustrated on the left below, we use a multicut as is illustrated on the right:

$$\dfrac{\Gamma \longrightarrow A \qquad \Gamma \longrightarrow G}{\Gamma \longrightarrow A \wedge G} \qquad\qquad \dfrac{\mathcal{P}; [\Gamma] \longrightarrow A \qquad \mathcal{P} \cup \{A\}; [\Gamma, A] \longrightarrow [A \wedge^+ G]}{\mathcal{P}; [\Gamma] \longrightarrow [A \wedge^+ G]}\,[mc.]$$

In this way, all attempts to prove A on the right will be bounded derivations of decide-depth of one.

One might consider that a table extracted from a proof is, in fact, a legitimate proof object since it is relatively easy to check that a table encodes a proof by first building a multi-cut derivation for it and then extending it to a proof. For example, within the proof carrying code framework [Necula 1997], it might be less expensive to transmit an ordered collection of atoms in order to represent a proof than to send some more complex representation of a sequent calculus proof tree. We will return to this aspect of tables in Section 3.11.

3.7.2 More than Horn clauses

The Horn clause subset described above is a natural class to consider since it involves only synchronous inference rules. Some asynchronous inference rules can also be accommodated in a similar fashion and this allows us to consider tabling in a richer setting than Horn clauses. In particular, consider allowing logic specifications (logic programs) to be the D-formula specified by the following grammar.

$$\begin{aligned} N &:= A \mid N_1 \wedge^- N_2 \mid G \supset N \mid \forall x\, N \\ G &:= true \mid A \mid G_1 \wedge^+ G_2 \mid G_1 \vee G_2 \mid \exists x\, G \mid \forall x\, G \mid D \supset G \\ D &:= A \mid N \mid D_1 \wedge^+ D_2 \mid \exists x\, D \end{aligned}$$

The class of all *first-order hereditary Harrop formulas* [Miller 1991] can be seen as being N-formulas as long as the positive occurrences of conjunctions in N are annotated as negative (that is, as $\wedge^-$). The notion of uniform proofs (goal-directed proofs) with respect to first-order hereditary Harrop formulas for this subset of intuitionistic logic corresponds to using LJF with all atoms given a negative polarity.

Since focused proofs are more general and flexible than uniform proofs, we will be able to consider the relationship between proof search and tabling in a larger subset of intuitionistic, with existential quantifiers and positive conjunctions on the left. Notice that these positive connectives appearing in D-formulas are decomposed in the asynchronous phase, by left-asynchronous rules, until there are only N-formulas present in the context. Moreover, all LJF^t_- positive trunks concluding a sequent focused on the left on a N-formula have the following form, modulo the order of the premises:

$$\frac{\mathcal{P}; [\Gamma] \xrightarrow{A} [R] \qquad \mathcal{P}; [\Gamma] -_{G_1}\!\!\rightarrow \qquad \cdots \qquad \mathcal{P}; [\Gamma] -_{G_n}\!\!\rightarrow}{\mathcal{P}; [\Gamma] \xrightarrow{N} [R]}$$

Here, $n \geq 0$, A is an atomic formula and $G_1, \ldots, G_n$ are G-formulas. From the figure above, it should be clear that if all atoms have negative polarity, then LJF^t_- proofs obtained from N-formula logic specifications and uniform proofs coincide: the left-most-premise must be the conclusion of an initial left rule, I_l, and R must be A. Therefore, these derivations correspond to the *backchaining steps* used in uniform proofs.

The restricted syntax for the formulas presented above translates to the following invariants regarding sequent occurrences within LJF^t_- proofs involving these formulas.

Lemma 3.17 *Let Ξ be an LJF^t_- proof of the sequent $\mathcal{P}; [\mathcal{P}, \Gamma], \Theta \longrightarrow G$, where Γ is a finite set of N-formulas, Θ is a finite set of D-formulas, $\mathcal{P}$ is a finite set of atoms and G is a G-formula. Sequents that occur in Ξ are of the following three kinds:*

1. *an unfocused sequent $\mathcal{P}'; [\mathcal{P}', \Gamma'], \Theta' \longrightarrow G'$ or $\mathcal{P}'; [\mathcal{P}', \Gamma'], \Theta' \longrightarrow [G'']$,*
2. *a right-focusing sequent $\mathcal{P}'; [\mathcal{P}', \Gamma'] -_{G'}\!\!\rightarrow$, or*
3. *a left-focusing sequent $\mathcal{P}'; [\mathcal{P}', \Gamma'] \xrightarrow{N} [G'']$,*

where $\mathcal{P}'$ is a set of atoms that contains $\mathcal{P}$, Γ' is a set of N-formulas that contains Γ, Θ' is a set of D-formulas (no relationship to Θ implied), G' is some G-formula, and G'' is a positive or atomic G-formula.

We show that if an atom is tabled, then the only possible proofs are those for which all the subproofs of this atom are bounded by a small number of decide rules.

Proposition 3.18 *Let Γ be a finite set of N-formulas, Θ and Θ' be finite sets of D-formulas, $\mathcal{P}$ be a set of atoms, G be a G-formula and $A \in \mathcal{P}$ be an atom. Let Ξ be a cut-free proof of the sequent $\mathcal{P}; [\mathcal{P}, \Gamma], \Theta \longrightarrow G$. Then all occurrences of the sequent $\mathcal{P}; [\Gamma'], \Theta' \longrightarrow A$ and $\mathcal{P}; [\Gamma'] -_A\!\!\rightarrow$ in Ξ are the conclusion of a derivation of decide-depth of at most one.*

Proof We again check all possible ways to prove the sequent $\mathcal{P}; [\Gamma'], \Theta' \longrightarrow A$. The other case, for the sequent $\mathcal{P}; [\Gamma'] -_A\!\!\rightarrow$, is similar.

The reasoning is close to the one done in the proof of the Proposition 3.15. Seeing from bottom up, one must apply asynchronous rules, decomposing Θ', until the asynchronous phase ends. Since no branching occurs in this phase, the resulting premise is of the form $\mathcal{P}; [\Gamma''] \longrightarrow [A]$, where Γ'' is a set of N-formulas containing Γ'. Now, there are two alternatives, either focus on the left, on a N-formula, or on the right. Let us examine the former case. If we focus on a N-formula on the left, then, as described above, the resulting synchronous derivation must contain a premise focused on the left on an atom. This premise is not a conclusion of any rule in LJF^t_- because A has positive polarity and the rule R_l is restricted to non-atomic formulas. Hence, there are no proofs by focusing on the left. Therefore, one must focus on the right and finish with an initial right rule. □

Before, in the Horn theory case, as the asynchronous phases of proofs were empty, there were no occurrences of the rules $\supset_r$, $\exists_l$, and $\forall_r$. Since these are the only rules that can introduce a new formula

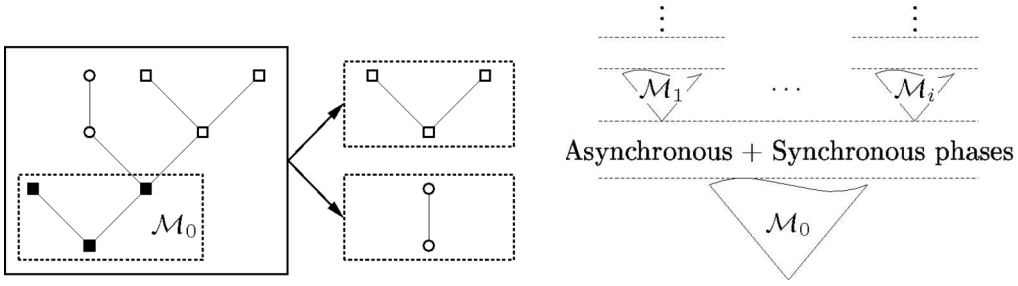


Figure 3.5: The left figure illustrates the algorithm that extracts a *tmcd* from an LJF proof tree. The right figure depicts the general architecture of a completed *tmcd* proof. The triangles represent *mcd*s and the dashed lines represents end of asynchronous phases.

or constant in the context of the sequent – the first rule introduces an hypothetical assumption, and the latter two, a new eigenvariable – we could assume that all elements of a table are provable from the bracketed context of the endsequent, and, therefore, insert a multicut derivation, constructed from a table, already on the bottom of the tree. Now with the case of logic specifications composed of hereditary Harrop formulas, the story is different. We cannot assume that all elements of a table are provable from the endsequent's context. This leads to the question: where to insert multicuts? Since we want to avoid reproving, it is better to insert multicut derivations as early as possible, that is, as early as the elements of a table are provable. So instead of extracting a table from a LJF proof, we will now extract a *tree of multicut derivations* (*tmcd*).

As its name suggests, a *tmcd* is a tree whose nodes are multicut derivations and whose branches connect one open premise of a multicut derivation to the endsequent of a sub-*tmcd*. Formally, a *tmcd* is specified as follows:

Definition 3.19 A *tmcd*, $\mathcal{T}$, is a finite set of one or more multicut derivations such that:

- there is a specially designated multicut derivation, $\mathcal{M}_{\mathcal{T}}$, called the root of $\mathcal{T}$; and
- the remaining multicut derivations can be partitioned in N possibly empty disjoint sets, $\mathcal{S}_i$, where N is the number of open premises in $\mathcal{M}_{\mathcal{T}}$, and such that
- for all $\mathcal{S}_i \neq \emptyset$, the elements of $\mathcal{S}_i$ can be further partitioned into $m \geq 1$ (ordered) subsets, $\mathcal{T}_1, \ldots, \mathcal{T}_m$, and each of these sets is a *tmcd*.

Intuitively, the first partition of the set specifies the edges in the *tmcd*, connecting one of the premises of the root multicut derivation and the endsequents of the *tmcd*s specified by the (second) partition of the subsets $\mathcal{S}_i$. The order of these subtrees specifies the branching order of the tree.

We now describe an algorithm to extract a *tmcd* from an LJF proof, Π, where all atoms are assigned negative polarity. We first pre-process Π by adding the empty polarity context to all of its sequents and transforming it into an LJF^t_- proof, Π'. For the algorithm, we only consider the sequents in Π' of the form $\mathcal{P}; [\Gamma] \longrightarrow [F]$. We use the Figure 3.5 to illustrate how this algorithm works, where the three different kinds of nodes (filled squares, ellipses, and blanked squares) denote sequents with different bracketed contexts. (1) We extract the maximal subtree, Ξ, at the bottom of the tree with the same bracketed context (in the example the subtree with filled squares) and construct from it the multicut derivation $\mathcal{M}_0$, by using a similar postorder traversal procedure described in the previous subsection, just that, since sequents have a polarity context, $\mathcal{P}$, representing tabled atoms, we do not table these atoms again. Assume that $\langle \mathcal{A}_0, \prec \rangle$ is the table of $\mathcal{M}_0$. Then, (2) by traversing Ξ,

we determine the children subtrees, $\Xi_1, \ldots \Xi_n$, of Ξ (in the example, the trees with blank squares and ellipses), and at the same time, we determine which sequent of the form $[\Gamma] \longrightarrow [F]$, where $F \in \mathcal{A}_0 \cup \{R\}$, is the closest to the endsequent of the each Ξ_i and the order, from left to right, a subtree appears, denoted by a natural number, j. At the end, we obtain tuples of the form $\langle j, F_i, \Xi_i \rangle$. In the example, we associate the same formula to the subtrees with ellipses and blank squares, but we associate to the former tree the order index 1 and to the latter subtree the order index 2. Now, (3) for each tuple $\langle j, F_i, \Xi_i \rangle$, we determine the set $\mathcal{P}_i = \{A \mid A \prec F_i\}$; (3.1) add this set to the bracketed context and polarity context of all the sequents in Ξ_i; and (3.2) replace all subderivations in Ξ_i concluding a sequent of the form $\mathcal{P}; [\Gamma_i] \longrightarrow [A]$, where $A \in \mathcal{P}_i$, by the derivation composed of the rules D_r and I_r, obtaining the derivation Ξ_i'. It is easy to check that Ξ_i' is still a valid LJF^t_- proof. Then, (4) we collect into a list all the subtrees Ξ_i' that have the same associated formula, F_i, taking into account the order of the associated order index. We obtain pairs of the form $(F_i, [\Xi_i^1, \ldots, \Xi_i^n])$. We now apply the algorithm recursively over the elements of the list. The formula F_i denotes the open premise of the *tmcd*'s root multicut derivation for which the *tmcd*s obtained from $\Xi_i^1, \ldots, \Xi_i^n$ are connected to.

Proposition 3.20 *Let Γ be a finite set of D-formulas, let G be a G-formula, and let Ξ be an LJF cut-free proof of $[\cdot]; \Gamma \longrightarrow G$, where all atoms are assigned negative polarity. Let Υ be the tree of multicut derivations obtained from Ξ by using the algorithm described before. Then Υ can be completed to a proof by adding derivations containing at most one occurrence of D_l rule.*

Proof By induction on the height of the tree of multicut derivations.

Base Case: Suppose that the height of Υ is 0, that is, there is only one multicut derivation, $\mathcal{M}$, in Υ. We prove the base case by another induction on the height of $\mathcal{M}$. The base case is trivial, since it would mean that the multicut $\mathcal{M}$ contains only one cut, and this cut would have to use an atom that is already present in the context of the proof. Therefore, it would require only one decide rule to complete the derivation to a proof.

Now the inductive step: Consider that the sequent $\mathcal{P}; [\Gamma \cup \mathcal{P}] \longrightarrow A_i$ is a premise of the multicut derivation $\mathcal{M}$. We can find a proof for this sequent with one decision left rule by proceeding as follows: We check in Ξ the formula F that was focused on to prove A_i. After this decision is made, one performs decide right-rules and decomposes the G-formula on the right-hand-side. Because of how the tables are constructed ($A \prec B$, then A is a subgoal of B), when these subgoals are completely decomposed, it must be the case that it encounters a previously proved atom, that is, a positive atom, and, therefore, another decision right finishes the proof with an initial right rule.

Inductive Step: We now have to show that there is a derivation between a branch sequent of a multicut derivation and its direct descendant multicut derivations that contains at most one decision left rule. This is similar to the base case. It suffices to decide in the same formula as in Ξ, and after performing a synchronous phase and later a possible asynchronous phase, there are two possible outcomes: 1) the formula on the right-hand-side is not a positive atom and, hence, there must be a descendant multicut for this goal; or 2) this formula is a positive atom, and, hence, a right decision rule is enough to finish the proof for this branch of the multicut derivation. In both cases, there is only one decision left rule. □

Notice that to find more efficiently the derivations that complete a *tmcd*, one also needs to agree on the names of the eigenvariables generated in the asynchronous phase. One possible way to do so is to consider the unbracketed context in the left-hand-side of sequents as lists, instead of multisets, similarly as with the asynchronous-sequents in LLF. Then, one could enumerate the eigenvariable according to the number of eigenvariables introduced in the path from where the variable is introduced to the endsequent.

3.8 A Proof Theoretic Notion for Fixed Points

The study of proof theoretic notions for fixed points dates back to Girard and Schroeder-Heister in [Girard 1992, Schroeder-Heister 1993]. Their initial motivation was to provide a proof theoretic explanation to *negation-as-failure* in the context of logic programming. We use the fixed point operator μ, in a similar way as in [Baelde 2007b, Baelde 2008], to specify fixed points.

The fixed point operator μ has type $(\tau \to \tau) \to \tau$, where τ is a type of the form $\gamma_1 \to \gamma_2 \to \cdots \gamma_n \to o$, each γ_i is a type of a term, n is the arity of the fixed point and o is the type of formulas. Its introduction rules are shown below:

$$\frac{\Gamma, B(\mu B)\vec{t} \longrightarrow C}{\Gamma, \mu B\vec{t} \longrightarrow C}\,[\mu_l] \qquad \frac{\Gamma \longrightarrow B(\mu B)\vec{t}}{\Gamma \longrightarrow \mu B\vec{t}}\,[\mu_r] \qquad \frac{}{\Gamma, \mu B\vec{t} \longrightarrow \mu B\vec{t}}\,[I_\mu]$$

We refer to the term B, in a fixed point $\mu B\vec{x}$, as the fixed point's *body* or *definition*. Its left and right introduction rules, above, just unfold the definition of a fixed point, while the third inference rule is the identity rule for fixed points.

Before we illustrate fixed points with some examples, we specify the proof theoretic approach to equality between terms, also due to Girard and Schroeder-Heister [Girard 1992, Schroeder-Heister 1993]. We will often need equalities in the body of fixed points. The left and right introduction rules of equality are as follows:

$$\frac{\{\Gamma\theta \longrightarrow \mathcal{R}\theta \mid \theta = mgu(s,t)\}}{\Gamma, t = s \longrightarrow \mathcal{R}}\,[=_l] \qquad \frac{}{\Gamma \longrightarrow s = s}\,[=_r]$$

While the right introduction rule is clear: one can introduce a equality composed of the same terms; the left introduction rule is more subtle. It contains a premise for each unifier θ of the terms s and t. Since we are dealing only with first-order logic terms, either this rule has only one premise if the terms s and t have a most general unifier, or it has no premise if s and t are not unifiable. Also notice that eigenvariables in the conclusion can also be substituted in its premise. We call the system obtained by adding to LJ the rules above for fixed points and equalities as LJ^μ.

Now we can represent logic programs as fixed points. Consider for example the program described in Section 3.2 that specifies when a node is reachable from another node. We represent this program by the fixed point below, denoted by *path*:

$$\mu(\lambda path \lambda x \lambda y.\ x = y \vee \exists z(adj\ x\ z \wedge path\ z\ y))$$

The left disjunct corresponds to the clause in the program stating that a node is connected to itself, and the right disjunct corresponds to the second clause of the program, with the recursion. The $adj\,x\,y$ predicate is also specified as a fixed point with definition $\bigvee_{(a,b)\in\mathcal{E}}(x = a \wedge^+ y = b)$, where $\mathcal{E}$ is the set of pairs with a graph's edges.

Consider again the graph in Figure 3.1, it is easy to show that the fixed point $path\ a_0\ a_5$ is provable. One just needs to unfold the fixed points on the right and chose the correct disjunct until one finishes the proof by applying right introduction rules for equality and conjunction, as illustrates the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\longrightarrow (a_0 = a_0 \wedge a_1 = a_1)}\,[\wedge_r, =_r, =_r]}{\longrightarrow (a_0 = a_0 \wedge a_1 = a_1) \vee (a_0 = a_1 \wedge a_1 = a_3) \vee \cdots}\,[n \times \vee_r]}{\longrightarrow adj\ a_0\ a_1}\,[\mu_r] \qquad \longrightarrow path\ a_1\ a_5}{\longrightarrow adj\ a_0\ a_1 \wedge path\ a_1\ a_5}\,[\wedge_r]}{\longrightarrow a_0 = a_5 \vee \exists z(adj\ a_0\ z \wedge path\ z\ a_5)}\,[\vee_r, \exists_r]}{\longrightarrow path\ a_0\ a_5}\,[\mu_r]$$

This derivation also illustrates that the right introduction rule for unfolding, together with the right introduction rule for equality, behaves closely to the backchaining rule in logic programming.

The main novelty of using fixed points and equalities lies on their left introduction rules. With these rules, we can prove, for example, the formula $\neg path\ a_1\ a_2$, which denotes that there is no path from the node a_1 and a_2. The proof follows by first applying an implication right rule, and then successive applications of the left introduction rules for fixed points, disjunctions and equality, as illustrates the following derivation.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{a_0 = a_5 \longrightarrow \bot}\,[=_l] \quad \dfrac{(a_0 = a_0 \wedge Z = a_1) \vee (a_0 = a_1 \wedge Z = a_3) \vee \cdots, path\ Z\ a_5 \longrightarrow \bot}{adj\ a_0\ Z, path\ Z\ a_5 \longrightarrow \bot}\,[\vee_l, \exists_l, \wedge_l]}{a_0 = a_5 \vee \exists z(adj\ a_0\ z \wedge path\ z\ a_5) \longrightarrow \bot}\,[\mu_l]}{path\ a_0\ a_5 \longrightarrow \bot}\,[\mu_l]}{\longrightarrow \neg path\ a_0\ a_5}\,[\supset_r]$$

By successive applications of the rules $\vee_l, \wedge_l$ and $=_l$, the right branch reduces to several premises where the eigenvariable Z is replaced by a node adjacent to a_1 and where one still needs to prove that there is no path from Z to a_2. The proof finishes by checking all possible paths in the graph that start from the node a_1. Notice that, as the size of this proof depends on the graph, this proof can be enormous. In practice, however, one uses tabling mechanisms to avoid proving twice the same formula, deriving, hence, smaller proof objects that do not contain redundant subproofs.

In order to preserve consistency, we assume that the body of fixed points are *monotonic*, that is, there are no negative recursive occurrences of the defined predicate. This is necessary for the termination of the cut-elimination algorithm for LJ$^\mu$. We do not show the cut-elimination theorem here, but, instead, we point out some other references, including works that prove this theorem for even more powerful logics, with induction and coinduction: McDowell & Miller in [McDowell 2000] proposed a proof system that also allows induction over natural numbers and proved that the cut-elimination theorem holds for this system. Tiu and Momigliano [Momigliano 2003, Tiu 2004] proposed the system LINC that includes rules for induction and coinduction. They also proved that the cut-elimination theorem holds for this logic. Later, Baelde and Miller [Baelde 2007b, Baelde 2008] investigated the focusing behaviors of systems with fixed points. Based on these logics, several proof assistants and tools have been proposed. Bedwyr [Baelde 2007a] is an implementation of a proof search strategy for a fragment of first-order logic with fixed points [Tiu 2005], which uses the rules above for introducing fixed points. We shall further explore this strategy in the next sections. Several (interactive) theorem provers that use similar notions of fixed points have also been proposed, for example, *Abella* [Gacek 2008] and *Tac* [Baelde 2008].

3.9 LJF$^\mu$

We now exploit the non-canonical focusing treatment of fixed points investigated by Baelde & Miller [Baelde 2007b, Baelde 2008]. They showed that there are complete focusing systems for linear logic with fixed points, where fixed points can be *frozen*, that is, their definition can no longer be unfolded and be treated as atoms. Thus, we can assign arbitrarily fixed points with negative or positive polarity. In particular, positive fixed points, when focused on, are used in initial rules, provided that there is a matching frozen fixed point in the context. In particular, we propose the intuitionistic system LJF$^\mu$, which allows fixed points to be assigned negative or positive polarity and is obtained by extending LJF in two different ways: first, we add to its sequents two declarations, $\mathcal{N}$ and $\mathcal{F}$, called the *polarity* and the *frozen* contexts. We consider that if a fixed point, $\mu B\vec{t}$, is an instance of an element in $\mathcal{N}$, denoted by the infix symbol $\in_\theta$, then it has negative polarity, otherwise it has positive polarity; and if $\mu B\vec{t} \in_\theta \mathcal{F}$ then $\mu B\vec{t}$ cannot be unfolded, but only be used in the initial rule. We classify these fixed points as *frozen*. They will play a major role to further control the shape of proofs. Second, we add

the rules for fixed points and equality shown in Figure 3.6. Initially, we assume that the polarity and frozen declarations are both empty, and they only change when a table is incorporated into the proof by using multicut rules (which will be introduced in the following sections).

As the *path* example above illustrates, by unfolding fixed points on the left, we can check if a property is satisfied in all paths. In fact, Tiu *et al.* in [Tiu 2005] propose a proof search strategy for proving G-formulas, specified in the grammar below, which consists on eagerly unfolding fixed points on the left. This strategy was implemented as a tool called Bedwyr [Baelde 2007a].

$$G ::= t \mid \bot \mid s = t \mid \mu(\lambda p \lambda \vec{x}.\, G)\vec{t} \mid G_1 \wedge^+ G_2 \mid G_1 \vee G_2 \mid P \supset G \mid \exists x.G \mid \forall x.G \mid A$$
$$P ::= t \mid \bot \mid s = t \mid \mu(\lambda p \lambda \vec{x}.\, P)\vec{t} \mid P_1 \wedge^+ P_2 \mid P_1 \vee P_2 \mid \exists x\, P \mid A$$

Notice that G-formulas appear always on the right-hand-side of the sequent, while P-formulas appear on the left-hand-side of the sequent. Moreover, P formulas contain only synchronous connectives, and, therefore, they are entirely decomposed in the asynchronous phase.

The proof strategy proposed by Tiu *et al.* is not complete because one never freezes a fixed point. Hence, if a fixed point that specifies an infinite fixed point definition, such as the natural numbers, is on the left-hand-side of the sequent, then one would unfold indefinitely many times. However, if we only consider *noetherian* definitions, that is, fixed point definitions whose unfoldings terminate, such as the graph example above, then we restore completeness. The proofs obtained through this proof search strategy correspond to the LJF$^{\mu}$ proofs that do not contain instances of initial rules I_l^{μ} and I_r^{μ}. In this chapter, we assume that all fixed points are *noetherian* and only consider the proofs of G-formulas above.

Proposition 3.21 *All asynchronous rules permute over all other rules, except the following pairs of rules:*

$$(\vee_l/\mu_r), \quad (\vee_l/\mu_l), \quad \textit{and} \quad (\wedge_r/\mu_l)$$

Proof Here we show some of the cases only:

- $\vee_r/\mu_l$:

$$\dfrac{\dfrac{\Gamma, B(\mu B)\vec{t} \longrightarrow P}{\Gamma, \mu B\vec{t} \longrightarrow P}\,[\mu_l]}{\Gamma, \mu B\vec{t} \longrightarrow P \vee Q}\,[\vee_r] \quad \rightsquigarrow \quad \dfrac{\dfrac{\Gamma, B(\mu B)\vec{t} \longrightarrow P}{\Gamma, B(\mu B)\vec{t} \longrightarrow P \vee Q}\,[\vee_r]}{\Gamma, \mu B\vec{t} \longrightarrow P \vee Q}\,[\mu_l]$$

- $\supset_l/\mu_l$:

$$\dfrac{\dfrac{\Gamma_1, B(\mu B)\vec{t} \longrightarrow P}{\Gamma_1, \mu B\vec{t} \longrightarrow P}\,[\mu_l] \quad \Gamma_2, Q \longrightarrow C}{\Gamma_1, \Gamma_2, P \supset Q, \mu B\vec{t} \longrightarrow C}\,[\supset_l] \quad \rightsquigarrow \quad \dfrac{\Gamma_1, B(\mu B)\vec{t} \longrightarrow P \quad \Gamma_2, Q \longrightarrow C}{\Gamma_1, \Gamma_2, P \supset Q, B(\mu B)\vec{t} \longrightarrow C}\,[\supset_l]}{\Gamma_1, \Gamma_2, P \supset Q, \mu B\vec{t} \longrightarrow C}\,[\mu_l]$$

- $\vee_r/=_l$:

$$\dfrac{\dfrac{\{\Gamma\theta \longrightarrow P\theta \mid \theta = mgu(s,t)\}}{\Gamma, s = t \longrightarrow P}\,[=_l]}{\Gamma, s = t \longrightarrow P \vee Q}\,[\vee_r] \quad \rightsquigarrow \quad \left\{ \dfrac{\dfrac{\Gamma\theta \longrightarrow P\theta}{\Gamma\theta \longrightarrow P\theta \vee Q\theta}\,[\vee_r] \quad \mid \theta = mgu(s,t)}{\Gamma, s = t \longrightarrow P \vee Q}\,[=_l] \right\}$$

- $\vee_r/=_l$:

$$\dfrac{\dfrac{\{\Gamma\theta \longrightarrow P\theta \mid \theta = mgu(s,t)\}}{\Gamma, s = t \longrightarrow P}\,[=_l]}{\Gamma, s = t \longrightarrow P \vee Q}\,[\vee_r] \quad \rightsquigarrow \quad \left\{ \dfrac{\dfrac{\Gamma\theta \longrightarrow P\theta}{\Gamma\theta \longrightarrow P\theta \vee Q\theta}\,[\vee_r] \quad \mid \theta = mgu(s,t)}{\Gamma, s = t \longrightarrow P \vee Q}\,[=_l] \right\}$$

□

The problem with the rules $\wedge_r$ and $\vee_l$ is that, in principle, one cannot ensure that a fixed point is unfolded in both premises. However, if this is the case, then the permutations work well, as illustrates the following transformation:

$$\dfrac{\dfrac{\Gamma, B(\mu B)\vec{t} \longrightarrow P}{\Gamma, \mu B\vec{t} \longrightarrow P}\,[\mu_l] \quad \dfrac{\Gamma, B(\mu B)\vec{t} \longrightarrow Q}{\Gamma, \mu B\vec{t} \longrightarrow Q}\,[\mu_l]}{\Gamma, \mu B\vec{t} \longrightarrow P \wedge Q}\,[\wedge_r] \quad \rightsquigarrow \quad \dfrac{\dfrac{\Gamma, B(\mu B)\vec{t} \longrightarrow P \quad \Gamma, B(\mu B)\vec{t} \longrightarrow Q}{\Gamma, B(\mu B)\vec{t} \longrightarrow P \wedge Q}\,[\wedge_r]}{\Gamma, \mu B\vec{t} \longrightarrow P \wedge Q}\,[\mu_l]$$

From the following proposition, proved by induction on the height of derivations, we can infer that the polarity and frozen contexts do not change in a cut-free proof. In the next section, we investigate how to manipulate these declarations by using multicut inference rules.

Proposition 3.22 *Let Ξ be an LJF$^\mu$ cut-free proof of $\mathcal{N}; \mathcal{F}; [\cdot] \longrightarrow G$, where G is a G-formula and $\mathcal{N}$ and $\mathcal{F}$ are declarations. Then all sequents in Ξ are of the form:*

$$\mathcal{N}; \mathcal{F}; [\Gamma] \xrightarrow{P} [G''] \qquad \mathcal{N}; \mathcal{F}; [\Gamma], \Delta \longrightarrow G' \qquad \mathcal{N}; \mathcal{F}; [\Gamma] \longrightarrow [G''] \qquad \text{or } \mathcal{N}; \mathcal{F}; [\Gamma] -_{G'} \rightarrow$$

where Γ is a set of P-fixed points and atoms, $\Delta \cup \{P\}$ is a set of P-formulas, G' is a G-formula and G'' is a positive or a fixpoint G-formula.

The soundness and completeness theorem for LJF$^\mu$ follows next.

Definition 3.23 If C is a G-formula or a P-formula, then $\hat{C}$ is the formula obtained by replacing all occurrences of $\wedge^+$ by $\wedge$. If Γ is a set of G-formulas and P-formula, then $\hat{\Gamma} = \{\hat{C} \mid C \in \Gamma\}$.

Theorem 3.24 *Let G be a G-formula and let $\mathcal{N}$ be an arbitrary polarity context. Assume all fixed points are* noetherian. *Then, the sequent $\longrightarrow \hat{G}$ is provable in LJ$^\mu$ if and only if the sequent $\mathcal{N}; \emptyset; [\cdot] \longrightarrow G$ is provable in LJ$^\mu$.*

Proof Soundness is straightforward, as it suffices to remove all focusing annotations.

For completeness, we just need to show that when a LJ$^\mu$ sequent, $\longrightarrow \hat{G}$, is provable in LJ$^\mu$, then the sequent that does not contain frozen fixed points $\mathcal{N}; \emptyset; [\cdot] \longrightarrow G$ is provable in LJF$^\mu$. Given that LJF is sound and complete with respect to LJ, we just need to consider the rules for the fixed points. The only interesting case is with the unfolding rule μ_l^a, as its principal formula is not contracted, but consumed in the premise. The proof is similar to the one in [McDowell 2003, Proposition 14]. We extend Girard's translation (see end of Section 2.4) with the translation of intuitionistic fixed points to linear fixed points $\ulcorner \mu P\vec{t} \urcorner \equiv \mu\, !\ulcorner P \urcorner \vec{t}$ and between equalities $\ulcorner s = t \urcorner \equiv s = t$. It is easy to check that this encoding is correct:

$$\dfrac{\Gamma, P(\mu P)\vec{t} \longrightarrow C}{\Gamma, \mu P\vec{t} \longrightarrow C} \quad \rightsquigarrow \quad \dfrac{\dfrac{!\ulcorner \Gamma \urcorner, !\ulcorner P \urcorner (\mu\, !\ulcorner P \urcorner)\vec{t} \vdash \ulcorner C \urcorner}{!\ulcorner \Gamma \urcorner, \mu\, !\ulcorner P \urcorner \vec{t} \vdash \ulcorner C \urcorner}}{!\ulcorner \Gamma \urcorner, !\,\mu\, !\ulcorner P \urcorner \vec{t} \vdash \ulcorner C \urcorner}$$

As pointed out by Schellinx [Schellinx 1991], faithfulness is more involved because of introduction rules that can close a premise whose right-hand-side has more than one formula, such as the left introduction rules for 0 and $=$. Adding fixed points, however, does not affect Schellinx's proof: unfolding a fixed point on the right yields a formula whose main connective is a !. Either this formula is erased by these introduction rules or the ! is introduced, which implies that the right-hand-side contains only one formula.

$$\frac{\mathcal{N};\mathcal{F};[\Gamma,\mu B\vec{t}],\Theta \longrightarrow \mathcal{R}}{\mathcal{N};\mathcal{F};[\Gamma],\Theta,\mu B\vec{t} \longrightarrow \mathcal{R}}\ [[]^\mu_l] \qquad \frac{\mathcal{N};\mathcal{F};[\Gamma],\Theta \longrightarrow [\mu B\vec{t}]}{\mathcal{N};\mathcal{F};[\Gamma],\Theta \longrightarrow \mu B\vec{t}}\ [[]^\mu_r]$$

$$\frac{\textbf{if } \mu B\vec{t} \in_\theta \mathcal{N}}{\mathcal{N};\mathcal{F};[\Gamma] \xrightarrow{\mu B\vec{t}} [\mu B\vec{t}]}\ [I^\mu_l] \qquad \frac{\textbf{if } \mu B\vec{t} \notin_\theta \mathcal{N}}{\mathcal{N};\mathcal{F};[\Gamma,\mu B\vec{t}]-\!\!\!\rightarrow_{\mu B\vec{t}}}\ [I^\mu_r]$$

$$\textbf{if } \mu B\vec{t} \notin_\theta \mathcal{F}$$

$$\frac{\mathcal{N};\mathcal{F};[\Gamma] \xrightarrow{B(\mu B)\vec{t}} [R]}{\mathcal{N};\mathcal{F};[\Gamma] \xrightarrow{\mu B\vec{t}} [R]}\ [\mu^s_l] \qquad \frac{\mathcal{N};\mathcal{F};[\Gamma],\Theta \longrightarrow B(\mu B)\vec{t}}{\mathcal{N};\mathcal{F};[\Gamma],\Theta \longrightarrow \mu B\vec{t}}\ [\mu^a_r]$$

$$\frac{\mathcal{N};\mathcal{F};[\Gamma],\Theta,B(\mu B)\vec{t} \longrightarrow \mathcal{R}}{\mathcal{N};\mathcal{F};[\Gamma],\Theta,\mu B\vec{t} \longrightarrow \mathcal{R}}\ [\mu^a_l] \qquad \frac{\mathcal{N};\mathcal{F};[\Gamma]-\!\!\!\rightarrow_{B(\mu B)\vec{t}}}{\mathcal{N};\mathcal{F};[\Gamma]-\!\!\!\rightarrow_{\mu B\vec{t}}}\ [\mu^s_r]$$

Equality Rules

$$\frac{\{\mathcal{N}\theta;\mathcal{F}\theta;[\Gamma\theta],\Theta\theta \longrightarrow \mathcal{R}\theta \mid \theta = mgu(s,t)\}}{\mathcal{N};\mathcal{F};[\Gamma],\Theta,t=s \longrightarrow \mathcal{R}}\ [=_l] \qquad \frac{}{\mathcal{N};\mathcal{F};[\Gamma]-\!\!\!\rightarrow_{s=s}}\ [=_r]$$

Figure 3.6: The rules in LJF$^\mu$ that introduce the fixed point operator, μ, and equality.

Now given this translation, the key observation is that, when fixed points are *noetherian* and they have P-formulas as body definitions, fixed points are *left-permeable*, that is, $\mu!^\ulcorner P^\urcorner\vec{t} \equiv !(\mu!^\ulcorner P^\urcorner\vec{t})$. This implies from cut-elimination that structural rules are admissible in the left-hand-side of the sequent:

$$\frac{\mu!^\ulcorner P^\urcorner\vec{t} \vdash !(\mu!^\ulcorner P^\urcorner\vec{t}) \qquad \Gamma,!(\mu!^\ulcorner P^\urcorner\vec{t}) \vdash C}{\Gamma,\mu!^\ulcorner P^\urcorner\vec{t} \vdash C}\ [\text{Cut}]$$

The proof of $\mu!^\ulcorner P^\urcorner\vec{t} \vdash !(\mu!^\ulcorner P^\urcorner\vec{t})$ relies on the fact that fixed points are *noetherian* and that a fixed point appearing on the left-hand-side of the sequent has a completely positive formula. We introduce eagerly the formulas in the left by applying asynchronous rules and unfolding fixed points until we obtain premises of the form $\vdash !(\mu!^\ulcorner P^\urcorner\vec{t}\theta)$, where θ is the substitution derived from applying $=_l$ rules. Now, to prove a particular premise, we just need to introduce the ! and simulate the path used to reach this premise from the endsequent, by applying the duals of the rules used in the path to reach this premise.

Finally, since unfolding of a fixed point can be done in either asynchronous or synchronous phase, the polarity assignment does not affect provability. □

The following proposition justifies the completeness of the proof search strategy adopted by Tiu *et al.* [Tiu 2005], which consists in not bracketing any fixed point, but always unfolding them.

Proposition 3.25 *Let G be a G-formula, and assume that all fixed points are* noetherian. *Then, for any polarity context $\mathcal{N}$, if the sequent $\mathcal{N};\emptyset;[\cdot],\cdot \longrightarrow G$ is provable in LJF$^\mu$, then there is an LJF$^\mu$ proof with no occurrences of $[]^\mu_l$.*

Proof We prove by induction on the number of $[]^\mu_l$ in an LJF$^\mu$ proof Ξ of $\mathcal{N};\emptyset;[\cdot],\cdot \longrightarrow G$. We

perform the following transformation:

$$\dfrac{\dfrac{\overset{\Xi}{[\Gamma, \mu P\vec{t}], \Delta \longrightarrow G}}{[\Gamma], \Delta, \mu P\vec{t} \longrightarrow G}\,[[]_l^\mu] \qquad \rightsquigarrow \qquad \dfrac{\overset{\Xi_1}{[\Gamma\theta_1], \Delta\theta_1 \longrightarrow G\theta_1} \quad \cdots \quad \overset{\Xi_n}{[\Gamma\theta_n], \Delta\theta_n \longrightarrow G\theta_n}}{[\Gamma], \Delta, \mu P\vec{t} \longrightarrow G}$$

where the polarity and freezing declarations are elided. Instead of applying $[]_l^\mu$ to the fixed point $\mu P\vec{t}$ we fully decompose it. This terminates because we are assuming that all fixed points are *noetherian* and P-formulas are completely asynchronous on the left. Ξ_i is obtained from Ξ by removing from the bracketed context the formula $\mu P\vec{t}$, applying the substitution θ_i, replacing all occurrences of initial rules involving $\mu P\vec{t}$ by the derivation where the right-hand-side is proved, similarly as done in the proof of Theorem 3.24, and finally, adapting the $=_l$ rules adequately with the correct unifiers. □

It is worth noticing that, because one can bracket a fixed point and use it in a initial rule, LJF$^\mu$ is more expressive than the logic behind Bedwyr. For example, one can prove the following statement about natural numbers: $\forall x(nat\, x \supset nat\,(s\, x))$, where *nat* is defined as the fixed point $\mu(\lambda nat \lambda x\,(x = z) \vee \exists x'(x = (s\, x') \wedge^+ nat\, x'))$. As one can unfold *nat* on the left indefinitely many times, one must bracket the occurrence of *nat* on the left and use it in an initial rule in order to prove this theorem.

3.10 A specific freezing discipline for tabling with fixed points

Proofs that contain fixed points can be enormous because they potentially contain redundant subproofs. For example, consider once more the graph in the Figure 3.1. If one attempts to prove the query $\neg path\ a_1\ a_2 \wedge^+ \neg path\ a_3\ a_2$ in a logic interpreter, then it would prove the subgoal $\neg path\ a_3\ a_2$ twice. As discussed before, since proving this literal involves checking all the paths in the graph that start with a_3, such proofs are as big as the graph itself. In implementations such as Bedwyr, one uses tabling mechanisms not only for atoms, but also for literals, to improve proof search performance. We will describe in the following subsection a freezing discipline for when fixed points and negated fixed points are tabled, and in the subsequent subsection, we extend this discipline for when universally quantified fixed points and negated fixed points are tabled.

3.10.1 Tables with fixed point literals

Instead of using polarity to denote membership of an atom in a table, we propose the following connection between freezing and tabling: when a fixed point is frozen it denotes that it is *decided* if this fixed point is positively or negatively proved. As before, we incorporate a table into a proof by using multicut rules, more specifically the mc_μ rule below. Here, the polarity of fixed points is not altered, but all of the instances of the introduced fixed points are frozen. We use the more general term *fixed point literal* for either a fixed point or its negation.

$$\dfrac{\mathcal{N}; \mathcal{F}; [\Gamma] \longrightarrow L_1 \quad \cdots \quad \mathcal{N}; \mathcal{F}; [\Gamma] \longrightarrow L_n \qquad \mathcal{N}; \mathcal{F} \cup \Theta_L; [\Gamma, \Delta_L] \longrightarrow [R]}{\mathcal{N}; \mathcal{F}; [\Gamma] \longrightarrow [R]}\,[mc_\mu]$$

where for all i, L_i is a fixed point literal, $\Delta_L = \{L_1, \ldots, L_n\}$, and $\Theta_L = \{F \mid F \in \Delta_L \text{ or } \neg F \in \Delta_L\}$, where F is a fixed point.

Notice that, since P-formulas can be constructed with disjunctions, we cannot guarantee that the proofs with interphase cuts are of the same size as the ones with intraphase cuts, as permuting cuts upwards might increase proofs. However, in all the examples considered, the elements of the tables are always provable from the context of the endsequent, which allows us to incorporate tables already at the bottom of the tree.

We consider that initially (at the endsequent) all fixed points are positive and not frozen, that is, the polarity context $\mathcal{N}$ and $\mathcal{F}$ are both empty. The polarity context does not play a major role in this subsection, but it will play a role in the next subsection, when we table universally quantified fixed point literals.

Although the following soundness and completeness result is only for when tabling P-fixed point formulas, we show, later in Section 3.12, some examples where one can also table G-fixed point formulas.

Proposition 3.26 *Let G be a G-formula and $\mathcal{F}$ be a set of P-fixed point-formulas, such that, for all $\mu P\vec{t} \in \mathcal{F}$, there is a proof of $\cdot \longrightarrow \mu\hat{P}\vec{t}$ in LJ$^\mu$. Assume all fixed points are* noetherian. *Then, for any polarity context $\mathcal{N}$, the sequent $\hat{\mathcal{F}} \longrightarrow \hat{G}$ is provable in LJ$^\mu$ if and only if the sequent $\mathcal{N}; \mathcal{F}; [\mathcal{F}] \longrightarrow G$ is provable in LJF$^\mu$.*

Proof Soundness is trivial, as it suffices to remove all focusing annotations.

From Proposition 3.25 and Theorem 3.24, we can assume that there is an LJF$^\mu$ proof for $\mathcal{N}; \emptyset; [\cdot], \mathcal{F} \longrightarrow G$ that does not contain instances of $[]^\mu_l$. We first remove all occurrences of unfoldings on the left-hand-side appearing in this proof by induction on the number of μ^a_l rules over frozen fixed points. As we assume that fixed points are always unfolded, we can permute rules on the left until a fixed point, $\mu P\vec{t}$, is completely decomposed last:

$$\dfrac{\dfrac{\overset{\Xi_1}{[\Gamma\theta_1], \Delta\theta_1 \longrightarrow G\theta_1} \quad \cdots \quad \overset{\Xi_n}{[\Gamma\theta_n], \Delta\theta_n \longrightarrow G\theta_n}}{[\Gamma], \Delta, P(\mu P)\vec{t} \longrightarrow G}}{[\Gamma], \Delta, \mu P\vec{t} \longrightarrow G}\ [\mu^a_l]$$

However since $\mu P\vec{t}$ is provable, it must be the case that at least one of the substitutions above, θ_i, is the identity substitution. Thus, we can replace the derivation above for the one that applies $[]^\mu_l$ over $\mu P\vec{t}$ and then complete the derivation with the proof obtained from $\Xi_i\theta_i$ by adding $\mu P\vec{t}$ to the bracketed context of all of its sequents. At the end, we will have an LJF$^\mu$ proof of $\mathcal{N}; \emptyset; [\mathcal{F}] \longrightarrow G$ that does not unfold any fixed point on the left. Now to remove the unfolding on the right, we just permute μ_r rules over asynchronous rules, so that all unfoldings on the right are done on sequents of the form $\mathcal{N}; \emptyset; [\mathcal{F}] \longrightarrow \mu P\vec{t}$, where $\mu P\vec{t} \in \mathcal{F}$. We then replace the derivation concluding these sequents by a derivation composed of $[]^\mu_r$ and an initial rule. Finally, we replace in all sequents the empty freezing context by $\mathcal{F}$. □

There is also a soundness and completeness result for when we table not only fixed points, but also negated fixed points. If $\neg\mu P\vec{t}$ is in the bracketed context, then one does not need to unfold an occurrence of $\mu P\vec{t}$ on the left because, as the context is inconsistent, one just needs to rely on this inconsistency (by focusing for example on $\neg\mu P\vec{t}$). Moreover, if there is an unfolding of $\mu P\vec{t}$ on the right, then it must be the case that both $\neg\mu P\vec{t}$ and $\mu P\vec{t}$ are provable, which implies that the context is again inconsistent.

By using the mc_μ rule above, we freeze in the right-most-branch the introduced fixed points, which on the other hand are proved in the left branches. This enforces that the only existing proofs for the right-most-premise are those that do not reprove finite failures or finite successes, denoted respectively by negated fixed points and fixed points. We start by showing the case for finite successes and then the case for finite failures.

Proposition 3.27 *Let Ξ be an arbitrary LJF$^\mu$ cut-free proof of the sequent $\mathcal{N}; \mathcal{F}; [\Gamma] \longrightarrow G$, where the fixed point $\mu P\vec{t} \in \mathcal{F} \cap \Gamma$ has positive polarity, $\mathcal{N} \subseteq \mathcal{F}$, Γ is a set of fixed points and G is a G-formula. Then every occurrence of the sequents $\mathcal{N}; \mathcal{F}; [\Gamma'] \longrightarrow \mu P\vec{t}$ or $\mathcal{N}; \mathcal{F}; [\Gamma']-_{\mu P\vec{t}}\!\!\rightarrow$ is the conclusion of a derivation of decide depth of at most one.*

Proof We show the case for when the sequent is $\mathcal{N}; \mathcal{F}; [\Gamma'] \longrightarrow \mu P\vec{t}$. The other case follows immediately from it. From Proposition 3.22 we can infer that it must be the case that Γ' is a set of fixed points. Now, we have to check all possible ways for which we can prove the sequent $\mathcal{N}; \mathcal{F}; [\Gamma'] \longrightarrow \mu P\vec{t}$. The first rule must be $[]^{\mu}_{r}$, since $\mu P\vec{t}$ is frozen. Then, one must prove the sequent $\mathcal{N}; \mathcal{F}; [\Gamma'] \longrightarrow [\mu P\vec{t}]$: if one focuses in the left, it must be the case that this fixed point has negative polarity. However, it fails immediately since one cannot apply neither the rule μ^{s}_{l} because $\mathcal{N} \subseteq \mathcal{F}$ and, hence, the fixed point is frozen; nor the rule I^{μ}_{l} since this fixed point is different from $\mu P\vec{t}$ because the latter has positive polarity, while the former has negative polarity; nor can we apply the R_l rule since the focused fixed point has negative polarity.

Now, the only remaining option is to focus on the right, which is possible since $\mu P\vec{t}$ has positive polarity. Then, we cannot apply the rule μ^{s}_{r} because $\mu P\vec{t}$ is frozen; nor can we apply the rule R_r because the fixed point is positive. The only remaining option is to apply I^{μ}_{r}. This succeeds because $\mu P\vec{t}$ belongs to Γ, and, therefore, it must be the case that it also belongs to Γ'. Hence, the only way to conclude this sequent is with a derivation with decide depth of at most one. □

Returning to our *path* example, we can now table *path* fixed points so that the only existing proofs are those that do not reprove the atom *path*, as illustrates the following derivation.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{[path\ a_3\ a_4] - _{adj\ a_1\ a_3} \rightarrow} \quad \dfrac{}{[path\ a_3\ a_4] - _{path\ a_3\ a_4} \rightarrow}\,[I^{\mu}_{r}]}{[path\ a_3\ a_4] - _{adj\ a_1\ a_3 \wedge^{+} path\ a_3\ a_4} \rightarrow}\,[\wedge^{+}_{r}]}{[path\ a_3\ a_4] - _{a_3 = a_4 \vee \exists z.adj\ a_1\ z \wedge^{+} path\ z\ a_4} \rightarrow}\,[\vee_r, \exists_r]}{[path\ a_3\ a_4] \longrightarrow a_3 = a_4 \vee \exists z.adj\ a_1\ z \wedge^{+} path\ z\ a_4}\,[[]_r, D_r]}{[path\ a_3\ a_4] \longrightarrow path\ a_1\ a_4}\,[\mu^{a}_{r}]$$

Here the polarity and frozen declarations are elided. Consider that the fixed point $path\ a_3\ a_4$ has positive polarity and is frozen, while the fixed point $path\ a_1\ a_4$ has positive polarity but is not frozen. Intuitively, the fixed point $path\ a_3\ a_4$ is introduced by a multicut rule mc_{μ}. We do not show the derivation introducing the sequent $[path\ a_3\ a_4] - _{adj\ a_1\ a_3} \rightarrow$, but one can easily check that it is a completely synchronous derivation obtained by selecting the correct disjunctions.

Notice, however, that, since asynchronous phases can be enormous, the number of decide rules in a derivation does not directly reflect the computational effort needed to find this derivation. In the path example, one still needs to check all possible paths. But nevertheless, performing a bounded proof search on the number of decide rules does reduce the amount of backtracking needed to search for a proof, reducing the overall computational effort. Moreover, as we show next, one can avoid that such asynchronous computations are performed twice by tabling negated fixed points.

Until now, there was no novelty, as we only dealt with the case when finite successes are tabled. We now proceed to the case when finite failures can also be tabled. We again use instances of mc_{μ} cut rules to introduce negated fixed points, $\neg\mu B\vec{t}$, into the proof. In the left branches, these formulas must be proved, while in the right-most-branch this formula is added to the bracketed context and all instances of these fixed points are frozen.

Lemma 3.28 *Let Ξ be an arbitrary LJF$^{\mu}$ proof of the sequent $\mathcal{N}; \emptyset; [\cdot] \longrightarrow G$, where G is a G-formula, and let $\mathcal{N}; \mathcal{F}; [\Gamma] \longrightarrow [R]$ be a sequent in Ξ. Then for all formulas $\neg\mu P\vec{t} \in \Gamma$, it must be the case that $\mu P\vec{t} \in \mathcal{F}$.*

Proof From Proposition 3.22, we can infer that, in any proof of a G-formula, the only way to introduce a negated fixed point to the bracketed context is by using an instance of the cut-rule mc_{μ}. Since once a fixed point is added to the frozen declaration it can no longer be removed, it must be the case that this fixed point is frozen. □

Proposition 3.29 *Let Ξ be an arbitrary LJF^{μ} cut-free proof of the sequent $\mathcal{N};\mathcal{F};[\Gamma] \longrightarrow G$, where the formula $\neg\mu P\vec{t} \in \Gamma$ and $\mu P\vec{t} \in \mathcal{F}$ has positive polarity, $\mathcal{N} \subseteq \mathcal{F}$, Γ is a set of fixed points literals, and G is a G-formula. Then every occurrence of the sequent $\mathcal{N};\mathcal{F};[\Gamma'] \longrightarrow \neg\mu P\vec{t}$ or $\mathcal{N};\mathcal{F};[\Gamma']-_{\neg\mu P\vec{t}}\!\!\rightarrow$ is the conclusion of a derivation of decide depth of at most one.*

Proof We proceed in a similar way as in Proposition 3.27, by checking all possible ways to conclude the sequent $\mathcal{N};\mathcal{F};[\Gamma'] \longrightarrow \neg\mu P\vec{t}$. The other case is similar.

The first rule must be $\supset_r$, followed by $[]^{\mu}_l, []_r$, because $\mu P\vec{t}$ is frozen. Now, one has to focus on the left. Since all negative fixed points on the left are frozen, they would not yield a proof, as we already discussed in Proposition 3.27. Therefore, one must focus on a negated fixed point, $\neg\mu P_2\vec{t}$, apply the rule $\supset_l$, resulting on premises of the form: $\mathcal{N};\mathcal{F};[\Gamma'] \xrightarrow{\perp} [\perp]$ and $\mathcal{N};\mathcal{F};[\Gamma']-_{\mu P_2\vec{t}}\!\!\rightarrow$. The former premise is then the conclusion of a derivation composed of the rules R_l and $false_l$. The latter premise can only be the conclusion of an initial rule, as, from Lemma 3.28, $\mu P_2\vec{t}$ must be frozen. This means that both $\neg\mu P_2\vec{t}$ and $\mu P_2\vec{t}$ belong to $\Gamma \cup \{\mu P\vec{t}\}$. Moreover, this derivation always exists because we could always choose to focus on the left on $\neg\mu P\vec{t}$ since $\mu P\vec{t}$ belongs to $\Gamma \cup \{\mu P\vec{t}\}$, for any Γ. □

We now are able to ensure, by using multicuts mc_{μ}, that the only proofs allowed are those for which a finite failure is not reproved. Let us return to the *path* example described before. Consider that we have introduced the negated literal $\neg path\ a_1\ a_2$ in the proof via a multicut. Then the proof of $\neg path\ a_0\ a_2$ has to be of the following shape, where the polarity and frozen declarations are elided and the fixed point $path\ a_1\ a_2$ is frozen and positive.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{[\neg path\ a_1\ a_2, path\ a_1\ a_2]-_{path\ a_1\ a_2}\!\!\rightarrow}\,[I^{\mu}_r] \quad \dfrac{\dfrac{}{[\neg path\ a_1\ a_2, path\ a_1\ a_2], \perp \longrightarrow \perp}\,[false_l]}{[\neg path\ a_1\ a_2, path\ a_1\ a_2] \xrightarrow{\perp} \perp}\,[R_l]}{[\neg path\ a_1\ a_2, path\ a_1\ a_2] \xrightarrow{\neg path\ a_1\ a_2} \perp}\,[\supset_l]}{[\neg path\ a_1\ a_2, path\ a_1\ a_2] \longrightarrow \perp}\,[D_l]}{[\neg path\ a_1\ a_2], path\ a_1\ a_2 \longrightarrow \perp}\,[[]^{\mu}_l]}{\begin{array}{c}\vdots\\ \cdots \quad [\neg path\ a_1\ a_2], adj\ a_0\ Z, path\ Z\ a_2 \longrightarrow \perp\end{array}}\,[\vee_l, \exists_l, \wedge^{+}_l]}{[\neg path\ a_1\ a_2], a_0 = a_2 \vee \exists z.adj\ a_0\ z \wedge^{+} path\ z\ a_2 \longrightarrow \perp}\,[\mu^{a}_l]}{[\neg path\ a_1\ a_2], path\ a_0\ a_2 \longrightarrow \perp}\,[\supset_r]}{[\neg path\ a_1\ a_2] \longrightarrow \neg path\ a_0\ a_2}$$

Here the horizontal ellipses represent the left branch of the $\vee_l$ rule, which contains only an instance of $=_l$. The vertical ellipses represent the derivation obtained by unfolding and decomposing the fixed point $adj\ a_0\ Z$, where Z is an eigenvariable.

Once we have already proved a finite failure, $\neg A$, whenever we encounter A as a subgoal in proof search, we would like to *fail quickly*, that is, disallow proof search to continue, or if the context is inconsistent, to finish the proof quickly. Because, whenever we introduce a finite failure with a cut rule, we freeze the fixed point constructing the literal, we can also simulate this behavior.

Proposition 3.30 *Let Ξ be an arbitrary cut-free (open) derivation of the sequent $\mathcal{N};\mathcal{F};[\Gamma] \longrightarrow G$, where the formula $\neg\mu P\vec{t} \in \Gamma$, $\mu P\vec{t} \in \mathcal{F}$ has positive polarity, $\mathcal{N} \subseteq \mathcal{F}$, Γ is a set of fixed points literals and G is a G-formula. Then every occurrence of the sequent $\mathcal{N};\mathcal{F};[\Gamma'] \longrightarrow \mu P\vec{t}$ or $\mathcal{N};\mathcal{F};[\Gamma']-_{\mu P\vec{t}}\!\!\rightarrow$ is the conclusion of a derivation of decide depth of at most one.*

Proof We again check all possible ways to conclude the sequent $\mathcal{N};\mathcal{F};[\Gamma'] \longrightarrow \mu P\vec{t}$. The other case is similar.

The first rule must be $[]_l^\mu$, as the rule μ_r^a is not applicable because the fixed point $\mu P\vec{t}$ is frozen. Now there are two options: focus on the right or on the left. If we focus on the right, then we can only proceed by completing the proof with an initial rule if $\mu P\vec{t} \in \Gamma$, since it has positive polarity and is frozen. This would imply that the context is inconsistent, as $\neg\mu P\vec{t}$ is also in Γ. If we decide on a formula in the left, then it must be the case that we decide on a negated fixed point, $\neg\mu B\vec{t}_2$, and not in a fixed point because $\mathcal{N} \subseteq \mathcal{F}$. One then applies the rule $\supset_l$. The resulting left premise is a sequent focused on a frozen positive fixed point which can only be the conclusion of a initial rule. The right premise is the conclusion of the closed derivation composed of R_l and $false_l$. In all cases, the derivation concluding the sequent $\mathcal{N}; \mathcal{F}; [\Gamma'] \longrightarrow \mu P\vec{t}$ has decide depth of at most one. □

Let us return to the third motivating example, described in Section 3.2, where an interpreter attempts to prove the goal $path\ a_2\ a_5$, but now we assume that $path$ is the fixed point described in Section 3.8 and that the formula $\neg path\ a_3\ a_5$ is tabled. If the interpreter attempts to prove the subgoal $path\ a_3\ a_5$, then it will not be allowed to traverse through all paths from a_3 to a_4, but it will fail immediately. This is illustrated by the following derivation where the polarity context, $\emptyset$, and the frozen context, $\{path\ a_3\ a_5\}$, are elided.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{[\neg path\ a_3\ a_5] - _{adj\ a_2\ a_3}\rightarrow} \quad \dfrac{}{[\neg path\ a_3\ a_5] - _{path\ a_3\ a_5}\rightarrow}\,[\wedge^+]}{[\neg path\ a_3\ a_5] - _{adj\ a_2\ a_3 \wedge^+ path\ a_3\ a_5}\rightarrow}\,[\vee_r, \exists_r]}{[\neg path\ a_3\ a_5] - _{a_2 = a_5 \vee \exists z.adj\ a_2\ z \wedge^+ path\ z\ a_5}\rightarrow}\,[[]_r, D_r]}{[\neg path\ a_3\ a_5] \longrightarrow a_2 = a_5 \vee \exists z.adj\ a_2\ z \wedge^+ path\ z\ a_5}\,[\mu_r^a]}{[\neg path\ a_3\ a_5] \longrightarrow path\ a_2\ a_5}$$

The interpreter cannot proceed proving the open premise as no rules are applicable: it cannot unfold $path\ a_3\ a_5$ because it is frozen; nor can it release focus because $path\ a_3\ a_5$ is positive; nor can it apply the initial rule because $path\ a_3\ a_5$ is not present in the context. Hence, the interpreter is forced to backtrack and instantiate the existential variables with the correct witnesses.

The extraction of tables with fixed point literals from proofs is not much more elaborated than the one described before for Horn clauses. Given an LJF$^\mu$ proof that does not contain $[]_l^\mu$, such as, the proofs obtained from Tiu *et al.*'s strategy, we table negated fixed points with the same postorder traversal algorithm, described in Subsection 3.7.1, but now we also include $\neg A$ in the partial order $\preceq'$, whenever we encounter sequents of the form $[\cdot]; A \longrightarrow \bot$.

3.10.2 Tables with universally quantified fixed point literals

Now, we go a step further and table not only fixed points literals, but also universally quantified fixed point literals which denote that every instance is either a finite success or a finite failure. For this we use the more general cut rule $mc_\mu^\forall$, shown below.

$$\dfrac{\mathcal{N}; \mathcal{F}; [\Gamma] \longrightarrow L_1 \quad \cdots \quad \mathcal{N}; \mathcal{F}; [\Gamma] \longrightarrow L_n \qquad \mathcal{N} \cup \Psi_L; \mathcal{F} \cup \Theta_L; [\Gamma, \Delta_L] \longrightarrow [R]}{\mathcal{N}; \mathcal{F}; [\Gamma] \longrightarrow [R]}\,[mc_\mu^\forall]$$

where for all i, L_i is a fixed point literal or a universally quantified fixed point literal, $\Delta_L = \{L_1, \ldots, L_n\}$, $\Psi_L = \{F \mid \forall\vec{x}.F \in \Delta_L\}$, and $\Theta_L = \{F \mid F \in \Delta_L \text{ or } \neg F \in \Delta_L \text{ or } \forall\vec{x}.F \in \Delta_L \text{ or } \forall\vec{x}.\neg F \in \Delta_L\}$, where F is a fixed point.

As before, when a fixed point is frozen, it denotes that it is *decided* if it is provable or it is not provable. One can show that this freezing discipline is sound and complete in a similar way as done in the previous subsection in Proposition 3.26: the argument is the same, just that we consider all instances of a tabled fixed point as frozen.

The polarity of fixed points plays an important role to ensure that the only existing proofs are those that always reprove a frozen fixed point with a bounded number of decide rules. This is done

by assigning the polarity of fixed points in such a way that, when the tabled formula is focused on, it is completely decomposed in the synchronous phase. Thus, when inserting into a proof a universally quantified fixed point, we change the polarity of all instances of the corresponding fixed points, so that when this formula is focused on the proof finishes immediately. Another consequence of such focusing discipline is that we cannot perform forward-chaining steps and, therefore, we do not have to restrict the reaction left rule R_l, as we did in Subsection 3.6.3.

Lemma 3.31 *Let Ξ be an arbitrary LJF$^\mu$ proof of $\emptyset;\emptyset;[\cdot] \longrightarrow G$, where G is a G-formula, and let $\mathcal{N};\mathcal{F};[\Gamma] \longrightarrow [R]$ be a sequent in Ξ. If $\forall\vec{x}.\mu B\vec{x} \in \Gamma$, then $\mu B\vec{x} \in \mathcal{N} \cap \mathcal{F}$ and if $\forall\vec{x}.\neg\mu B\vec{x} \in \Gamma$, then $\mu B\vec{x} \in \mathcal{F}$.*

Proof The proof is similar to the proof of Lemma 3.28. □

Notice that, from the lemma above, we cannot always guarantee that all instances of the fixed point constructing a universally quantified negated fixed point have positive polarity. It can happen that, in a proof, the polarity of some instances are changed to negative by using an instance of the cut-rule $mc^{\forall}_{\mu}$ with a universally quantified fixed point, changing the focusing behavior of these instances and, thus, changing the focusing behavior of universally quantified negated fixed points in the context. However, if this is the case, it means that the table we used to construct the multicut derivation is not consistent, as it contains formulas whose instance and of its negation are both provable. Here, we assume that tables are consistent. In fact, we only need the weaker condition, stated below, that sequents are *polarized consistently*, which implies that the tables used do not contain universally quantified formulas of fixed point, A, and of a negated fixed point, $\neg B$, for which A and B are unifiable.

Definition 3.32 A sequent $\mathcal{N};\mathcal{F};[\Gamma] \longrightarrow G$ is *polarized consistently* if and only if for all formulas of the form $\forall\vec{x}.\neg\mu B\vec{x} \in \Gamma$, all instances of $\mu B\vec{x}$ have positive polarity and for all formulas of the form $\forall\vec{x}.\mu B\vec{x} \in \Gamma$, all instances of $\mu B\vec{x}$ have negative polarity.

Lemma 3.33 *Let Ξ be an LJF$^\mu$ cut free proof of a polarized consistently sequent $\mathcal{N};\mathcal{F};[\Gamma] \longrightarrow G$, where G is a G-formula and Γ is a set of fixed points literals and universally quantified fixed point literals. Then, every sequent in Ξ is also polarized consistently.*

Proof Simple proof by induction on the height of proofs. Since Ξ is cut-free and only cut-rules can modify the polarity and frozen contexts and insert universally quantified fixed point literals in the context, it must be the case that all sequents in Ξ are polarized consistently. □

As before, the only possible proofs are those where reproving a finite success and a finite failure is bounded by derivations of decide-depth one.

Proposition 3.34 *Let Ξ be an LJF$^\mu$ cut free proof of a polarized consistently sequent $\mathcal{N};\mathcal{F};[\Gamma] \longrightarrow G$, where G is a G-formula, $\forall\vec{x}.\mu B\vec{x} \in \Gamma$, $\mu B\vec{x} \in \mathcal{N} \cap \mathcal{F}$, $\mathcal{N} \subseteq \mathcal{F}$ and Γ is a set of fixed points literals and universally quantified fixed point literals. Then every occurrence of the sequents $\mathcal{N};\mathcal{F};[\Gamma'] \longrightarrow \mu B\vec{t}$ or $\mathcal{N};\mathcal{F};[\Gamma']-_{\mu B\vec{t}}\!\!\longrightarrow$ is the conclusion of a derivation of decide-depth of at most one.*

Proof We again check all the derivations that can conclude the sequent $\mathcal{N};\mathcal{F};[\Gamma'] \longrightarrow \mu B\vec{t}$. The other case is similar.

The first rule is $[]^{\mu}_{r}$, as the rule μ^{a}_{r} is not applicable because $\mu B\vec{x} \in \mathcal{N} \cap \mathcal{F}$. Then we must decide on a formula on the left, as all instances of $\mu B\vec{x}$ are negative. One then can either focus on a negated fixed point or a universally quantified fixed point literal. For the cases when we decide on a negated fixed point or a universally quantified negated fixed points, the resulting synchronous derivation will contain as premise a sequent focused on a frozen and positive fixed point on the right, which can only be the conclusion of an initial rule, and a sequent focused on *false* on the left, which is the conclusion

of a derivation composed of the rules R_l and $false_l$. Now the case for when we focus on a universally quantified fixed point only succeeds if we focus on a formula that can be instantiated by applying $\exists_l$ rules to $\mu B\vec{t}$, such as, deciding on the formula $\forall\vec{x}.\mu B\vec{x} \in \Gamma$ because from Lemma 3.33 the sequent is polarized consistently. In all cases, the derivation that concludes the sequent $\mathcal{N}; \mathcal{F}; [\Gamma'] \longrightarrow \mu B\vec{t}$ has decide-depth of at most one. □

Proposition 3.35 *Let Ξ be an LJF^{μ} cut free proof of a polarized consistently sequent $\mathcal{N}; \mathcal{F}; [\Gamma] \longrightarrow G$, where G is a G-formula, $\forall\vec{x}.\neg\mu B\vec{x} \in \Gamma$, $\mu B\vec{x} \in \mathcal{F}$, $\mathcal{N} \subseteq \mathcal{F}$ and Γ is a set of fixed points literals and universally quantified fixed point literals. Then every occurrence of the sequents $\mathcal{N}; \mathcal{F}; [\Gamma'] \longrightarrow \neg\mu B\vec{t}$ or $\mathcal{N}; \mathcal{F}; [\Gamma']-_{\neg\mu B\vec{t}}\!\!\rightarrow$ or $\mathcal{N}; \mathcal{F}; [\Gamma'] \longrightarrow \mu B\vec{t}$ or $\mathcal{N}; \mathcal{F}; [\Gamma']-_{\mu B\vec{t}}\!\!\rightarrow$ is the conclusion of a derivation of decide-depth of at most one.*

Proof The proof is a similar to the proofs of Propositions 3.29, 3.30, and 3.34. □

We now return to the problem of finding the elements of a table. In particular, we now describe a method to determine how to universally quantify a *fixed point*, $\mu B\vec{x}$. This method uses the *logical variables* introduced by a logic interpreter like Prolog. Logic interpreters use logical variables to postpone the decision of providing witnesses for the bounded variable in the $\exists_r$ and $\forall_l$ rules. The decision is made later by solving a unification problem, which yields a substitution for the logical variables. Applying such substitution to the proof object returned by the interpreter yields a valid proof in the logic. For example, if we use the logic program encoding the path example described before and ask the query $\exists x\, path\, x\, a_3$, then the interpreter would replace the existential variable x by a new logical variable X and return a substitution for it. In this case, there are three proof objects where the X is replaced by a_0, a_1 or a_3, respectively.

Since interpreters find the most-general-unifiers, it might be the case that the substitution found by the interpreter replaces a logical variable, X, with a term containing other logical variables $\vec{X}$. In these cases, the logical variables, $\vec{X}$, behave like eigenvariables, that is, one can instantiate, in the proof obtained by the interpreter, any variable in $\vec{X}$ by any term, and the resulting object is still a valid proof. Hence, to determine how to universally quantify a fixed point, $\mu B\vec{x}$, we ask the query $\exists\vec{x}\mu B\vec{x}$ to an interpreter, obtaining a substitution θ. Then, we can safely universally quantify over the logical variables, $\vec{X}$, appearing in $(\mu B\vec{x})\theta$.

Notice however, that we cannot use the same method to determine how to universally quantify *negated fixed points*, $\neg\mu B\vec{x}$, as these fixed points make use of equalities on the left. It is not yet clear how to proceed when these equalities contain logical variables. Consider, for example, the query $\exists x.(x = 0 \supset \bot)$. The interpreter would have to find a substitution for x that does not satisfy the unification problem $x = 0$, which is a problem of a different nature. Bedwyr, for example, does not allow logical variables to appear in the left-hand-side of the sequent.

3.11 Table as proof object

We have illustrated how tables can be incorporated into proofs. To what extent can we think of tables as proofs themselves? Of course, this question is best addressed when one knows what one will do with a proof.

In the *proof carrying code* setting [Necula 1997], proof objects are transmitted together with mobile codes to assure that some (safety) properties are satisfied by these programs. Before a client executes the transmitted code the client checks that the proof that the code is carrying proves the program's safety. Thus, proof objects must be engineered so that they are not too large (in order to reduce transmission costs) and not too complex to check (in order to reduce resource requirements on client proof checkers).

Tables might well be a good format for proofs in this setting for several reasons. First, tables represent declarative information and not procedural information: in particular, tables only describe

what is provable and do not go into detail about how things are proved. Proof checking can then be organized around simple proof search engines that implement, for example, LJF^t_-. The trade-offs between proof size and proof checking time are fairly clear: if the producer of a proof tables all successfully proved atoms (as in Proposition 3.16), then tables can be large but proof checking can be simple (only proofs of decide-depth 1 must be considered in extending a multicut derivation). On the other hand, if some atomic formulas are not tabled, then the client may have to reprove them: clearly, reproving some atomic formulas might be rather straightforward and something that a client might be willing to do to help reduce the size of a transmitted proof.

In [Roychoudhury 2000], Roychoudhury *et.al.* propose using tables to build *justifications* that can be seen as a kind of proof. In their setting, these proof objects serve to explain why a logic program can or cannot prove a given atom. They argue that their justification can be used within model checkers and parsers. It seems likely that our use of tables as proofs can be used in these settings as well.

In the next section, we describe some tables that represent proof objects.

3.12 Examples

Example 3.36 We now return to the example of the Fibonacci sequence. Let the table, $\mathcal{T}$, contain the Fibonacci sequence up to the N^{th} Fibonacci number, denoted as $\mathit{fib}(N, \mathit{fib}_n)$, and the partial order, $\preceq$, be such that $\mathit{fib}(X,Y) \preceq \mathit{fib}(X+1,Z)$ for all $\mathit{fib}(X,Y) \in \mathcal{T}$. Moreover, let $\mathcal{M}$ be the multicut derivation obtained from $\mathcal{T}$ and used to prove that $\mathit{fib}(N, \mathit{fib}_n)$. Then, $\mathcal{M}$ has a unique proof that is linear in size. This contrasts with the previous cases obtained by changing the polarity of *fib* atoms: either there are infinitely many proofs, where the smallest is linear in size, or there is a unique proof, but it is exponential in size.

Example 3.37 Similarly to [McDowell 2003], we use the fixed point definition *step* below, that specifies a transition from the state P to the state Q by performing the action A:

$$\begin{array}{l}\mu(\lambda step \lambda P \lambda A \lambda Q.\\ \quad \exists P'.[P = A.P' \wedge^+ Q = A.P'] \vee\\ \quad \exists P' Q' R.[P = (P' \mid R) \wedge^+ Q = (Q' \mid R) \wedge^+ step\, P'\, A\, Q'] \vee\\ \quad \exists P' Q' R.[P = (R \mid P') \wedge^+ Q = (R \mid Q') \wedge^+ step\, P'\, A\, Q'] \vee\\ \quad \exists P' R.[P = (P' + R) \wedge^+ (step\, P'\, A\, Q \vee step\, R\, A\, Q)])\end{array}$$

where $A.P$, $P + Q$ and $P \mid Q$ represent a process prefixed by an action A, the choice, and the parallel composition, respectively. Although we only show a fragment of the concurrent language CCS [Milner 1989], larger fragments could also be easily accommodated, as in [McDowell 2003]. Notice that the fixed points *step* are completely positive, thus if focused on the right, these fixed points are fully decomposed.

McDowell *et.al.* showed in [McDowell 2003] that the fixed point definition *sim*, defined below,

$$\mu(\lambda sim \lambda p \lambda q. \forall A \forall P' step\, p\, A\, P' \supset [\exists Q'.step\, q\, A\, Q' \wedge^+ sim\, P'\, Q'])$$

can be used to compute the simulation relation. In particular, a process P is simulated by a process Q if and only if the atomic formula $sim\, P\, Q$ is provable. (Bisimulation can be encoded using a slightly more complex definition.) Moreover, if Ξ is a cut-free proof of that atomic formula and if $\mathcal{S}$ is the set of all pairs $\langle t, s\rangle$ such that Ξ contains a subproof of $sim\, t\, s$, then $\mathcal{S}$ is a simulation. Furthermore, let $\preceq$ be the post-order relation on $\mathcal{S}$ derived from Ξ as described in Section 3.7. Notice that it is now a simple matter to check that $\mathcal{S}$ is, in fact, a simulation by treating it as a table and considering extending its induced multicut derivation to a complete proof. In particular, let $\langle p, q\rangle \in \mathcal{S}$ and let $\mathcal{F} = \{sim\, t\, s \mid \langle t, s\rangle \in \mathcal{S}, \text{ and } sim\, t\, s \preceq sim\, p\, q\}$. An attempt to extend the sequent

$\emptyset; \mathcal{F}; [\mathcal{F}] \longrightarrow sim\, p\, q$ yields a proof of the form, where the polarity and freezing declarations are elided:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{[\mathcal{F}] - step\ q\ a\ q' \rightarrow} \quad \dfrac{}{[\mathcal{F}] - sim\ p'\ q' \rightarrow}\,[I^{\mu}_{r}]}{[\mathcal{F}] - step\ q\ a\ q' \wedge^{+} sim\ p'\ q' \rightarrow}\,[\wedge^{+}_{r}]}{\cdots \quad [\mathcal{F}] \longrightarrow [\exists Q'.\ step\ q\ a\ Q' \wedge^{+} sim\ p'\ Q']}\,[D_r, \exists_r] \quad \cdots}{\vdots}$$

$$\dfrac{\dfrac{}{[\mathcal{F}], step\ p\ A\ P' \longrightarrow [\exists Q'.\ step\ q\ A\ Q' \wedge^{+} sim\ P'\ Q']}}{[\mathcal{F}] \longrightarrow \forall A, P'.\ step\ p\ A\ P' \supset \exists Q'.\ step\ q\ A\ Q' \wedge^{+} sim\ P'\ Q'}\,[2 \times \forall_r, \supset_r, []_r]$$

The vertical ellipses represent the asynchronous phase obtained by unfolding and decomposing the fixed point $step\, p\, A\, P'$, and the horizontal ellipses represent the other premises resulting from this unfolding. At the end of this asynchronous phase, the eigenvariables A and P' are replaced by the action a and the state p'. Notice that there is only one D_r rule in this proof and that if the fixed point $step\, p\, A\, P'$ is frozen, instead of unfolded, the goal is no longer provable because, for any instance of Q', one cannot prove the fixed point $step\, q\, A\, Q'$.

Example 3.38 Consider a game between two players, named 1 and 2, who alternate in playing (consider tic-tac-toe) and that one player wins when the other player cannot move. We assume that the state of the game is encoded as a term in the logic and that the fixed point $move\, P\, Q$ encodes the fact that there is move from position P to Q (similar to the *adj* fixed point). Furthermore, assume that there are no infinite plays. Then there is a winning strategy for player 1 from the position P if and only if the fixed point $win\, P$, where *win* is the fixed point definition

$$\mu(\lambda win \lambda P. \forall P'\ move\, P\, P' \supset \exists Q\ [move\, P'\, Q \wedge^{+} win\, Q])$$

As with the previous example, let Ξ be a proof of the fixed point $win\, p$, let $\mathcal{W}$ be the set of atoms of the form $win\, P$ that are proved in subproofs of Ξ, and let $\preceq$ be the post-order traversal ordering of $\mathcal{W}$ based on Ξ. It is now a simple matter to verify that $\mathcal{W}$ encodes a winning strategy: simply build the multicut derivation associated to the table $\mathcal{W}$ and extend it to a complete proof. This later step is essentially the same kind of restricted proof search that is presented for the previous example based on simulation.

Example 3.39 Consider a simple functional programming language, with abstractions, applications and let constructors. We use the following fixed point definition, denoted by *of*, to typecheck programs in this programming language:

$$\begin{array}{l}
\mu(\lambda\, of \lambda\, \Gamma_a \lambda\, \Gamma_l \lambda\, U \lambda\, T. \\
\quad [member\ (U, T)\ \Gamma_a] \vee \\
\quad [member\ (U, M)\ \Gamma_l \wedge^{+} of\, \Gamma_a\, \Gamma_l\, M\, T] \vee \\
\quad \exists R, T_1, T_2. [U = abs\, R \wedge^{+} T = arr\, T_1\, T_2 \wedge^{+} \forall X. of\, ((X, T_1) :: \Gamma_a)\, \Gamma_l\, (R\, X)\, T_2] \vee \\
\quad \exists M, N, T_1. [U = app\, M\, N \wedge^{+} of\, \Gamma_a\, \Gamma_l\, M\, (arr\, T_1\, T) \wedge^{+} of\, \Gamma_a\, \Gamma_l\, N\, T_1] \vee \\
\quad \exists M, N, T_1. [U = (let\, M\, N) \wedge^{+} \\
\qquad \forall X. (of\, \Gamma_a\, (X, M) :: \Gamma_l\, X\, T_1 \wedge^{+} of\, \Gamma_a\, (X, M) :: \Gamma_l\, (N\, X)\, T)])
\end{array}$$

The fixpoint $of\, \Gamma_a\, \Gamma_l\, U\, T$ denotes that the term U has type T in the contexts Γ_a and Γ_l. Terms are constructed by using the function symbols: *abs* of type $(term \rightarrow term) \rightarrow term$, denoting abstractions; *app* of type $term \rightarrow term \rightarrow term$, denoting application of terms; and *let* of type $term \rightarrow (term \rightarrow term) \rightarrow term$, denoting *let* terms. Types are constructed by using the function symbol *arr*, denoting function types. The context Γ_a is a list of pairs, (X, T), where the first element, X, is an eigenvariable and the second element, T, is a term denoting the type of the variable X. Intuitively, these variables denote the new variables in the context that are introduced when typechecking abstractions. Finally,

the context Γ_l is also a list of pairs, (X, M), where X is an eigenvariable and M a term, which stores the let binding variables together with the term they represent.

The first disjunct, in the definition of the fixed point *of*, checks if the pair (U, T) is a member of the list Γ_a, denoted by the fixed point *member*, which is specified by the usual fixed point used to check membership in a list. The second disjunct checks if the pair (U, M) is in the context Γ_l, which means that we are typechecking a let binding variable associated to the term M. If this is the case, then one checks if the type of M is T. The third and fourth disjuncts correspond, respectively, to the typechecking of an abstraction and of an application, where :: is the usual list constructor. Notice that, for typechecking an abstraction, one must create a new eigenvariable X and insert it together with its type into the Γ_a context. Finally, the fifth disjunct is used to typecheck *let* programs of the form *let* M N. Notice that when this formula is focused on, then focus is lost and the asynchronous phase starts by performing the rule $\forall_r$, which creates a new eigenvariable for the let binder and adds it together with the term M in the context Γ_l. Then one checks if X has a type T_1 and if the term $N\ X$ has type T.

Although one can use the fixed point *of* to typecheck the types of programs, its performance is impractical as the cut-free proofs obtained from it might contain too many redundancies. In fact, the great success of traditional typechecking algorithms, such as the one proposed by Milner [Milner 1978], is that it avoids such redundancies by using *polymorphic types* containing type variables that can be instantiated to any type. For example, consider the following program *let* $\lambda x.x\ N$. Instead of substituting all occurrences of the binder of N by the term $\lambda x.x$, one finds the most general type for $\lambda x.x$, which is $\forall\alpha.\alpha \to \alpha$. Then, one continues by checking the type of Nf, assuming that the type of f is $\forall\alpha.\alpha \to \alpha$. Whenever needed, one instantiates the type of f, by instantiating the type variable α.

In our setting, polymorphic types are specified by formulas of the form:

$$\forall\vec{\alpha}, \Gamma'_a, \Gamma'_l.of\ (\Gamma'_a :: \Gamma_a)\ (\Gamma'_l :: \Gamma_l)\ U\ T$$

denoting that the type of U is T for any instance of the type variables $\vec{\alpha}$ and any increments, Γ'_a and Γ'_l, to the contexts Γ_a and Γ_l. To enforce that these formulas are used and not reproved, we use the multicut rule $mc^{\forall}_{\mu}$ to introduce them into the proof before we check the type of the terms, M and N, in a let term. Consider, for example, the proof of the sequent $\emptyset; \emptyset; [\cdot] \longrightarrow of\ \Gamma_a\ \Gamma_l\ (let\ m\ n)\ t$. One unfolds the fixed point, then focuses on the right-hand-side, selects the last disjunct, by successive applications of $\vee_r$ rule, instantiates the existential variables and decomposes the positive conjunction. We obtain two premises: one focused on the equality $let\ m\ n = let\ m\ n$, which ends with $=_r$, and the other premise is the conclusion of the the following derivation, where the polarity and frozen contexts are both empty and are elided:

$$\dfrac{\dfrac{[\cdot] \longrightarrow [of\ \Gamma_a\ (X, m) :: \Gamma_l\ X\ t_1 \wedge^+ of\ \Gamma_a\ (X, m) :: \Gamma_l\ (n\ X)\ t)]}{[\cdot] \longrightarrow \forall X.(of\ \Gamma_a\ (X, m) :: \Gamma_l\ X\ t_1 \wedge^+ of\ \Gamma_a\ (X, m) :: \Gamma_l\ (n\ X)\ t)}\ [\forall_r, []_r]}{[\cdot] -_{\forall X.(of\ \Gamma_a\ (X,m)::\Gamma_l\ X\ t_1 \wedge^+ of\ \Gamma_a\ (X,m)::\Gamma_l\ (n\ X)\ t)} \longrightarrow}\ [R_r]$$

At this point, we use the multicut rule $mc^{\forall}_{\mu}$ to introduce into the proof the formula $F = \forall\vec{\alpha}, \Gamma'_a, \Gamma'_l.of\ (\Gamma'_a :: \Gamma_a)\ (\Gamma'_l :: (X, m) :: \Gamma_l)\ X\ T_\alpha$, denoting the polymorphic type for the term m. As discussed at the end of Subsection 3.10.2, we can use a logic programming interpreter to determine which are the universally quantified variables in F[1]. We can use the same proof obtained from this interpreter to prove the left-premise of the cut. Now the right-premise of the cut rule must be

[1] In fact, in order to obtain the correct answer set substitution, one needs an extra term, say *var*, to denote both let and abstraction variables. More specifically, one needs to replace, in the third and fifth disjuncts, all occurrences of universally quantified variables, X, by *var* X, and in the first two disjuncts, one must also check that the term being typechecked is a variable, *var* U.

the conclusion of the following derivation, where the polarity and frozen contexts contain only the formula $of\,(\Gamma'_a :: \Gamma_a)\,(\Gamma'_l :: (X, m) :: \Gamma_l)\,X\,T_\alpha$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{}\,[3 \times \forall_l, I_l^\mu]}{[F] \xrightarrow{F} [of\,\Gamma_a\,\Gamma_l^x\,X\,t_1]}\,[D_l]}{[F] \longrightarrow [of\,\Gamma_a\,\Gamma_l^x\,X\,t_1]}\,[R_r, []_r^\mu \star] \quad \dfrac{\Xi}{[F]{-_{of\,\Gamma_a\,\Gamma_l^x\,(n\,X)\,t)}\!\!\longrightarrow}}}{[F]{-_{of\,\Gamma_a\,\Gamma_l^x\,X\,t_1}\!\!\longrightarrow}}}{[F]{-_{of\,\Gamma_a\,\Gamma_l^x\,X\,t_1 \wedge^+ of\,\Gamma_a\,\Gamma_l^x\,(n\,X)\,t)}\!\!\longrightarrow}}\,[\wedge_r^+]}{[F] \longrightarrow [of\,\Gamma_a\,\Gamma_l^x\,X\,t_1 \wedge^+ of\,\Gamma_a\,\Gamma_l^x\,(n\,X)\,t)]}\,[D_r]$$

where $\Gamma_l^x = (X, m) :: \Gamma_l$. Notice that in $\star$ one has to release focus because all instances of the fixed point $of\,(\Gamma'_a :: \Gamma_a)\,(\Gamma'_l :: (X, m) :: \Gamma_l)\,X\,T_\alpha$ are negative. Moreover, in Ξ all attempts to check the type of X will be the conclusion of a similar derivation as the derivation above of the left-premise.

3.13 Conclusions and future works

In this chapter, we investigated how to give a declarative answer to the question of reusing and not reproving previously proved lemmas. We proposed two focusing disciplines for tabled formulas which ensured that the only existing proofs are those that reuse tabled formulas. The first focusing discipline consisted in assigning positive polarity to tabled atoms. It was then applied in some fragments of first-order-logic, namely when logic specifications are hereditary Harrop formulas and when tables contained only atomic formulas. The second focusing discipline consisted in freezing fixed points that are tabled. It was applied to small fragments of first-order-logic, where all fixed points are *noetherian* and when tables contained only fixed point literals and universally quantified fixed point literals. Finally, we illustrated their use with some examples, such as simulation of two processes, winning strategies and let polymorphism typechecking.

We now point out some future work directions:

- *Tabling more than universally quantified fixed points* – One could imagine tabling formulas that have implications, such as formula of the form $\forall\vec{x}[(\mu B_1\vec{x} \wedge^+ \ldots \wedge^+ \mu B_n\vec{x}) \supset \mu B\vec{x}]$. In this case, if we consider the focusing discipline where all/some instances of the formula $\mu B\vec{x}$ are frozen and with negative polarity, then one could ensure that the lemma above is used whenever there is an instance of $\mu B\vec{x}$ on the right-hand-side of the sequent, obtaining derivations of the form:

$$\dfrac{[\Gamma]{-_{\mu B_1\vec{t}}\!\!\longrightarrow} \qquad \cdots \qquad [\Gamma]{-_{\mu B_n\vec{t}}\!\!\longrightarrow}}{[\Gamma] \longrightarrow [\mu B\vec{t}]}$$

  The challenge is to find the conditions for which such focusing discipline is complete and how to find or extract such tabled formulas.

- *Tables as proof objects* – Here, we have just speculated that one could use tables as proof objects in the proof carrying code framework. One still has to understand the trade-offs between the amount of information in the table and the "work" necessary for completing a table to a proof. This is also connected to the process of *cut-introduction* that consists in introducing cuts into a (cut-free) proof to obtain smaller proofs.

- *Not only* noetherian *fixed points* – Here, we have always assumed that all fixed points are *noetherian*. It seems possible to generalize some of the results obtained here to the more general setting. The first task is to prove the completeness of systems that consume fixed points whenever they are unfolded in the asynchronous phase. For this one has to show that

by only contracting positive formulas the contraction of fixed points is admissible. One could try, for example, to encode LJ^{μ} in linear logic in similar way as in [Liang 2008].

Tabling such fixed points is also related to avoiding cyclic proofs, which is another important use of tabling systems. For example, consider that we have a graph that contains cycles. If we attempt to prove that a node a is not connected to another node b, then it might happen that, while searching for a proof, the interpreter encounters a cycle in the graph. This would generate a cycle in proof search. Systems like Bedwyr [Baelde 2007a] use tabling mechanisms to find such cycles, and depending if the cycle is with a greatest or a least fixed point the proof search fails or succeeds.

Chapter 4

A framework for proof systems

Meta-logics and type systems based on intuitionistic logic are commonly used for specifying natural deduction proof systems. We shall show here that linear logic can be used as a meta-logic to specify a range of object-level proof systems. In particular, we show that by providing different *polarizations* within a *focused proof system* for linear logic, one can account for natural deduction (normal and non-normal), sequent proofs (with and without cut), and tableaux proofs. Armed with just a few, simple variations to the linear logic encodings, more proof systems can be accommodated, including proof systems using generalized elimination and generalized introduction rules. In general, most of these proof systems are developed for both classical and intuitionistic logics. By using simple results about linear logic, we can also give simple and modular proofs of the soundness and relative completeness of all the proof systems we consider.

References: parts of this chapter appeared in the conference paper [Nigam 2008b] and appeared in the extended and improved version of this paper [Nigam 2009b].

4.1 Introduction

Logics and type systems have been exploited in recent years as frameworks for the specification of deduction in a number of logics. The most common such *meta-logics* and *logical frameworks* have been based on intuitionistic logic (see, for example, [Felty 1988, Paulson 1989]) or dependent types (see [Harper 1993, Pfenning 1989]). Such intuitionistic logics can be used to directly encode natural deduction style proof systems.

In a series of papers [Miller 1996, Pimentel 2001, Miller 2002, Miller 2004, Pimentel 2005], Miller & Pimentel used classical linear logic as a meta-logic to specify and reason about a variety of sequent calculus proof systems. Since the encodings of such logical systems are natural and direct, the meta-theory of linear logic can be used to draw conclusions about the object-level proof systems. For example, in [Miller 2002], a decision procedure was presented for determining if one encoded proof system is derivable from another. In the same paper, necessary conditions were presented (together with a decision procedure) for assuring that an encoded proof system satisfies cut-elimination. This last result used linear logic's dualities to formalize the fact that if the left and right introduction rules are suitable duals of each other then non-atomic cuts can be eliminated.

In this chapter, we again use linear logic as a meta-logic but make critical use of the completeness of *focused proofs* for linear logic. In particular, we exploit the fact that literals can be assigned with arbitrary polarity and that different polarity assignments may yield different linear logic proofs. The earlier works of Miller & Pimentel assumed that all atoms were given negative polarity: this assignment resulted in an encoding of object-level sequent calculus. As we shall show here, if we vary that polarity assignment, we can get other object-level proof systems represented. Thus, while provability is not affected, different meta-level focused proofs are built and these encode different object-level proof systems.

Our main contribution in this chapter is illustrating how a range of proof systems can be seen as different focusing disciplines on the same or (meta-logically) equivalent sets of linear logic specifications. Soundness and relative completeness of the encoded proof systems are generally derived

via simple arguments about the structure of linear logic proofs. In particular, we present examples based on sequent calculus and natural deduction [Gentzen 1969], Generalized Elimination Rules [von Plato 2001], Free Deduction [Parigot 1992], the tableaux system KE [D'Agostino 1994], and Smullyan's Analytic Cut [Smullyan 1968a]. The adequacy of a given specification of inference rules requires first assigning polarity to meta-level literals used in the specification: then adequacy is generally an immediate consequence of the focusing theorem of linear logic.

Comparing two proof systems can be done at three different levels of "adequacy": *relative completeness* claims simply that the provable sets of formulas are the same, *full completeness of proofs* claims that the completed proofs are in one-to-one correspondence, and *full completeness of derivations* claims that (open) derivations (such as inference rules themselves) are also in one-to-one correspondence. All the proof systems that we shall encode will be done with this third, most refined level of adequacy.

This chapter is structured as follows: in Section 4.2, we describe how object-level formulas and sequents are encoded in linear logic, and we introduce the three levels of adequacy for encodings mentioned above. In Section 4.3 we show how to encode sequent calculus systems for minimal, intuitionistic and classical logic. Then, in Sections 4.4 and 4.5, we encode natural deduction systems for minimal and intuitionistic logics, with and without generalized elimination rules. Section 4.6 shows the encoding of Parigot's Free Deduction system [Parigot 1992] for classical logic. In Sections 4.7 and 4.8, we encode the tableaux systems for propositional classical logic proposed by D'agostino *et al.* [D'Agostino 1994] and Smullyan [Smullyan 1968a]. Finally in Sections 4.9 and 4.10, we end this chapter by commenting some related works and pointing out some future work directions.

4.2 Preliminaries

4.2.1 Encoding object-logic formulas and proof contexts

The proof systems that we encode have (partial) proofs that involve formulas in two *sense*. For example, in the process of building a natural deduction proof, some formulas are hypothesis (one argues *from* such formulas) and some formulas are conclusions (one argues *to* such formulas). In the process of building a sequent calculus proofs, some formulas are on the left of the sequent arrow and some are on the right. Tableaux proofs similarly use signed formulas (with either a **T** or **F** sign [Smullyan 1968b]) or places formulas on the left or right of a turnstile [D'Agostino 1994].

Informally, we will think of a *proof context* as being a collection of object-level formulas that are each present in these two senses. Thus, when encoding natural deduction, this collection can be a set or a multiset of object-level formulas marked as either being an hypothesis or the conclusion. In order to provide a consistent presentation of proof contexts throughout the range of proof systems, we introduce the two meta-level predicates $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ of type $form \to o$: the meta-level atomic formulas $\lfloor B\rfloor$ and $\lceil B\rceil$ are then used to denote these two different senses of how the object-level formula B is used within a proof context. The meta-level focused sequent $\vdash \Theta : \Gamma \Uparrow \cdot$ can then be used to collect together atomic formulas into a set via the unbounded context Θ or into a multiset via the bounded context Γ. Thus, the object-level sequent $B_1, \ldots, B_n \vdash C_1, \ldots, C_m$ can be encoded as the LLF sequent $\vdash \cdot : \lfloor B_1\rfloor, \ldots, \lfloor B_n\rfloor, \lceil C_1\rceil, \ldots, \lceil C_m\rceil \Uparrow \cdot$ if both the left and right side of the object-level sequent are multisets. If, say, the left side is a set and the right side is a multiset, then this sequent could be represented as $\vdash \lfloor B_1\rfloor, \ldots, \lfloor B_n\rfloor : \lceil C_1\rceil, \ldots, \lceil C_m\rceil \Uparrow \cdot$. Here, formulas on the left of the object-level sequent are marked using $\lfloor\cdot\rfloor$ and formulas on the right of the object-level sequent are marked using $\lceil\cdot\rceil$. For convenience, if Γ is a (multi)set of formulas, $\lfloor\Gamma\rfloor$ (resp. $\lceil\Gamma\rceil$) denotes the multiset of atoms $\{\lfloor F\rfloor \mid F \in \Gamma\}$ (resp. $\{\lceil F\rceil \mid F \in \Gamma\}$).

The theory $\mathcal{L}$ given in Figure 4.1 will be used throughout this chapter in order to axiomatize the two senses for all the connectives in both intuitionistic and classical logic. For example, the conjunction connective appears in two formulas: once in the scope of $\lfloor\cdot\rfloor$ and once in the scope of

$$
\begin{array}{llll}
(\Rightarrow_L) & \lfloor A \Rightarrow B \rfloor^\perp \otimes (\lceil A \rceil \otimes \lfloor B \rfloor) & (\Rightarrow_R) & \lceil A \Rightarrow B \rceil^\perp \otimes (\lfloor A \rfloor \parr \lceil B \rceil) \\
(\wedge_L) & \lfloor A \wedge B \rfloor^\perp \otimes (\lfloor A \rfloor \oplus \lfloor B \rfloor) & (\wedge_R) & \lceil A \wedge B \rceil^\perp \otimes (\lceil A \rceil \,\&\, \lceil B \rceil) \\
(\vee_L) & \lfloor A \vee B \rfloor^\perp \otimes (\lfloor A \rfloor \,\&\, \lfloor B \rfloor) & (\vee_R) & \lceil A \vee B \rceil^\perp \otimes (\lceil A \rceil \oplus \lceil B \rceil) \\
(\forall_L) & \lfloor \forall B \rfloor^\perp \otimes \lfloor Bx \rfloor & (\forall_R) & \lceil \forall B \rceil^\perp \otimes \forall x \lceil Bx \rceil \\
(\exists_L) & \lfloor \exists B \rfloor^\perp \otimes \forall x \lfloor Bx \rfloor & (\exists_R) & \lceil \exists B \rceil^\perp \otimes \lceil Bx \rceil \\
(\bot_L) & \lfloor \bot \rfloor^\perp & (t_R) & \lceil t \rceil^\perp \otimes \top
\end{array}
$$

Figure 4.1: The theory $\mathcal{L}$ used to encode various proof systems for minimal, intuitionistic, and classical logics.

$$
\begin{array}{llllll}
(Id_1) & \lfloor B \rfloor^\perp \otimes \lceil B \rceil^\perp & (Id_2) & \lfloor B \rfloor \otimes \lceil B \rceil & (Id_2') & \lfloor B \rfloor \otimes !\lceil B \rceil \\
(Str_L) & \lfloor B \rfloor^\perp \otimes ?\lfloor B \rfloor & (Str_R) & \lceil B \rceil^\perp \otimes ?\lceil B \rceil & (W_R) & \lceil C \rceil^\perp \otimes \bot
\end{array}
$$

Figure 4.2: Specification of the identity rules (cut and initial) and of the structural rules (weakening and contraction).

$\lceil \cdot \rceil$. Notice that this axiomatization is independent of the proof systems that this theory is used to describe. When we display formulas in this manner, we intend that the named formula is actually the result of applying ? to the existential closure of the formula. Thus, the formula named $(\wedge_L)$ is actually $?\exists A \exists B [\lfloor A \wedge B \rfloor^\perp \otimes (\lfloor A \rfloor \oplus \lfloor B \rfloor)]$. Furthermore, for intuitionistic and minimal logics, we use, instead, the two following formulas for the meaning of the implication:

$$
(\supset_L) \quad \lfloor A \supset B \rfloor^\perp \otimes (!\lceil A \rceil \otimes \lfloor B \rfloor) \qquad (\supset_R) \quad \lceil A \supset B \rceil^\perp \otimes (\lfloor A \rfloor \parr \lceil B \rceil)
$$

The bang in the formula $(\supset_L)$ will be important to correctly encode the structural restriction for these logics, where sequents contain at most one formula in their right-hand-side. We denote by $\mathcal{L}_J$ the set obtained from $\mathcal{L}$ by replacing the formulas $(\Rightarrow_L)$ and $(\Rightarrow_R)$ by $(\supset_L)$ and $(\supset_R)$, and we denote by $\mathcal{L}_M$ the set obtained by removing the formula $(\bot_L)$ from $\mathcal{L}_J$.

The formulas in Figure 4.2 also play a central role in presenting proof systems. The Id_1 and Id_2 formulas can prove the duality of the $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ predicates: in particular, one can prove in linear logic that

$$
\vdash \forall B(\lceil B \rceil \equiv \lfloor B \rfloor^\perp) \,\&\, \forall B(\lfloor B \rfloor \equiv \lceil B \rceil^\perp), Id_1, Id_2
$$

These two formulas are used, for example, to encode the initial and cut rules when we shall encode object-level sequent calculi (Section 4.3). To correctly encode the structural restrictions of intuitionistic and minimal logics, however, we use the clause Id_2', instead of Id_2. Notice that one can no longer prove the equalities above from the formulas Id_1 and Id_2', and hence the dualities between $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ formulas do not hold in minimal and intuitionistic logics.

The formulas Str_L and Str_R allow us to prove the equivalences $\lfloor B \rfloor \equiv ?\lfloor B \rfloor$ and $\lceil B \rceil \equiv ?\lceil B \rceil$. The last two equivalences allow the weakening and contraction of formulas at both the meta-level and object-level. For instance, in the encoding of minimal logics, where structural rules are only allowed in the left-hand-side, one should include only the Str_L formula; while in the encoding of classical logics, where structural rules are allowed in both sides of a sequent, one should include both Str_L and Str_R formulas. The formula W_R encodes the weakening right rule and is used to encode intuitionistic logics, where weakening, but not contraction, is allowed on formulas on the right-hand-side of a sequent.

From the Str_L clause we can derive the equivalence $\lfloor B \rfloor^\perp \equiv !\lfloor B \rfloor^\perp$ by negating the equivalence $\lfloor B \rfloor \equiv ?\lfloor B \rfloor$ obtained from this clause. This equivalence allows us to insert the ! before negative occurrences of $\lfloor \cdot \rfloor$. The presence of bangs in theories will play an important role in encoding correctly the structural rules of logics, such as minimal and intuitionistic logics, which require that right-hand-

sides of sequents do not contain more than one formula. Although these equivalences do not affect provability, applying them can change focusing behavior significantly.

4.2.2 Adequacy levels for encodings

When comparing deductive systems, one can easily identify several "levels of adequacy." For example, Girard in [Girard 2006, Chapter 7] proposes three levels of adequacy based on semantical notions: the level of *truth* (level -1), the level of *functions* (level -2), and the level of *actions* (level -3). Here, we also identify three levels of adequacy but from a proof-theoretical point-of-view. The weakest level of adequacy is *relative completeness* (level -1) which considers only *provability*: a formula has a proof in one system if it has a proof in another system. A stronger level of adequacy is of *full completeness of proofs* (level -2): the proofs of a given formula are in one-to-one correspondence with proofs in another system. We can consider an even stronger level of adequacy. We use the term *full completeness of derivations* (level -3) if the derivations (such as inference rule themselves) in one system are in one-to-one correspondence with those in another system.

For each of the object-logic proof systems that we consider here, we propose a meta-level theory, say $\mathcal{L}'$, that can be used to encode that system at the strongest level of adequacy. In all cases, we obtain $\mathcal{L}'$ from the formulas in Figures 4.1 and 4.2 by some combination of the following steps.

1) **Applying equivalences.** As we have shown, some equivalences are derivable from the identity and structural rules. Hence, we will at times replace occurrences of, for example, $\lfloor F \rfloor^{\perp}$ with $\lceil F \rceil$.

2) **Incorporating structural rules into introduction rules.** Although the formulas Str_L and Str_R provide an elegant specification of the weakening and contraction structural rules for the two difference senses for object-level formulas, they do not provide a good focusing behavior since the equivalences they imply can yield loops in a specification. Therefore, we incorporate the structural rules into a theory by adding ? and ! in its formulas. This transformation to a theory is usually formally justified using an induction of the height of proofs.

3) **Switching between multiplicative and additive introduction rules.** Given the presence of ? and ! within the specification of inference rules and the linear logic equivalences $?(A \oplus B) \equiv ?A \parr ?B$ and $!(A \,\&\, B) \equiv !\,A \otimes !\,B$ it is possible to replace, for example, the "additive" version of the rules $(\wedge_L), (\wedge_R), (\vee_L), (\vee_R)$ in $\mathcal{L}$ with their "multiplicative" version, namely with

$$\begin{array}{ll} \lceil A \wedge B \rceil^{\perp} \otimes (\lceil A \rceil \otimes \lceil B \rceil) & \lfloor A \wedge B \rfloor^{\perp} \otimes (\lfloor A \rfloor \parr \lfloor B \rfloor) \\ \lfloor A \vee B \rfloor^{\perp} \otimes (\lceil A \rceil \otimes \lceil B \rceil) & \lceil A \vee B \rceil^{\perp} \otimes (\lceil A \rceil \parr \lfloor B \rfloor). \end{array}$$

Formal justification of this step will also be done using an induction on the height of proofs.

When we build $\mathcal{L}'$ from $\mathcal{L}$ and the rules in Figure 4.2 based on these steps, it will be a simple matter to prove that the new theory $\mathcal{L}'$ proves exactly the same formulas as the original theory. However, before we can formally say that a theory $\mathcal{L}'$ describes a proof system, we must assign polarity to the meta-level atomic formulas $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$. Only then can we claim that *the "macro-rules" that result from focusing on formulas in that theory match exactly the inference rules of the corresponding encoded object-logic proof system.* This polarity assignment may differ between different proof system encodings.

Although we concentrate on obtaining encodings of proof systems at the highest levels of adequacy, it is worth noticing that one might still be interested in theories that are adequate only at the level of (complete) proofs. For example, following the Curry-Howard isomorphism, functional programs are complete proofs and their execution involves the construction of cut-free proofs out of these programs. In that domain, one may not require adequacy at the level of (open) derivations.

$$\frac{\Gamma, A \supset B \vdash A \quad \Gamma, A \supset B, B \vdash C}{\Gamma, A \supset B \vdash C}\ [\supset L] \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B}\ [\supset R]$$

$$\frac{\Gamma, A_1 \wedge A_2, A_i \vdash C}{\Gamma, A_1 \wedge A_2 \vdash C}\ [\wedge Li] \qquad \frac{\Gamma \vdash A, \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ [\wedge R]$$

$$\frac{\Gamma, A \vee B, A \vdash C \quad \Gamma, A \vee B, B \vdash C}{\Gamma, A \vee B \vdash \Delta}\ [\vee L] \qquad \frac{\Gamma \vdash A_i}{\Gamma \vdash A_1 \vee A_2}\ [\vee R_i]$$

$$\frac{\Gamma, \forall x\, A, A\{t/x\} \vdash C}{\Gamma, \forall x\, A \vdash C}\ [\forall L] \qquad \frac{\Gamma \vdash A\{c/x\}}{\Gamma \vdash \forall x\, A}\ [\forall R]$$

$$\frac{\Gamma, \exists x\, A, A\{c/x\} \vdash C}{\Gamma, \exists x\, A \vdash C}\ [\exists L] \qquad \frac{\Gamma \vdash A\{t/x\}}{\Gamma \vdash \exists x\, A}\ [\exists R]$$

$$\frac{\Gamma, A \vdash C \quad \Gamma \vdash A}{\Gamma \vdash C}\ [\text{Cut}] \qquad \frac{}{\Gamma, A \vdash A}\ [\text{I}] \qquad \frac{}{\Gamma \vdash t}\ [tR]$$

Figure 4.3: The sequent calculus, LM^{c}, for minimal logic. Here, c is not free in $\Gamma \cup \{C\}$ and $i \in \{1, 2\}$.

$$\frac{}{\Gamma, \bot \vdash \cdot}\ [\bot L] \qquad \frac{\Gamma \vdash \cdot}{\Gamma \vdash C}\ [\text{WR}]$$

Figure 4.4: The rules to add to LM^{c} to obtain the sequent calculus, LJ^{c}, for intuitionistic logic.

$$\frac{\Gamma, A \Rightarrow B \vdash A, \Delta \quad \Gamma, A \Rightarrow B, B \vdash \Delta}{\Gamma, A \Rightarrow B \vdash \Delta}\ [\Rightarrow L] \qquad \frac{\Gamma, A \vdash A \Rightarrow B, B, \Delta}{\Gamma \vdash A \Rightarrow B, \Delta}\ [\Rightarrow R]$$

$$\frac{\Gamma, A_1 \wedge A_2, A_i \vdash \Delta}{\Gamma, A_1 \wedge A_2 \vdash \Delta}\ [\wedge Li] \qquad \frac{\Gamma \vdash A \wedge B, A, \Delta \quad \Gamma \vdash A \wedge B, B, \Delta}{\Gamma \vdash A \wedge B, \Delta}\ [\wedge R]$$

$$\frac{\Gamma, A \vee B, A \vdash \Delta \quad \Gamma, A \vee B, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta}\ [\vee L] \qquad \frac{\Gamma \vdash A_1 \vee A_2, A_i, \Delta}{\Gamma \vdash A_1 \vee A_2, \Delta}\ [\vee R_i]$$

$$\frac{\Gamma, \forall x\, A, A\{t/x\} \vdash \Delta}{\Gamma, \forall x\, A \vdash \Delta}\ [\forall L] \qquad \frac{\Gamma \vdash \forall x\, A, A\{c/x\}, \Delta}{\Gamma \vdash \forall x\, A, \Delta}\ [\forall R]$$

$$\frac{\Gamma, \exists x\, A, A\{c/x\} \vdash \Delta}{\Gamma, \exists x\, A \vdash \Delta}\ [\exists L] \qquad \frac{\Gamma \vdash \exists x\, A, A\{t/x\}, \Delta}{\Gamma \vdash \exists x\, A, \Delta}\ [\exists R]$$

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma \vdash A, \Delta}{\Gamma \vdash \Delta}\ [\text{Cut}] \qquad \frac{}{\Gamma, A \vdash A, \Delta}\ [\text{I}] \qquad \frac{}{\Gamma \vdash t, \Delta}\ [tR] \qquad \frac{}{\Gamma, \bot \vdash \Delta}\ [\bot L]$$

Figure 4.5: The sequent calculus, LK^{c}, for classical logic. Here, c is not free in $\Gamma \cup \{C\}$ and $i \in \{1, 2\}$.

$(\supset_L)$	$\lfloor A \supset B \rfloor^\perp \otimes (!\lceil A \rceil \otimes ?\lfloor B \rfloor)$	$(\supset_R)$	$\lceil A \supset B \rceil^\perp \otimes (?\lfloor A \rfloor \parr \lceil B \rceil)$
$(\wedge_L)$	$\lfloor A \wedge B \rfloor^\perp \otimes (?\lfloor A \rfloor \oplus ?\lfloor B \rfloor)$	$(\wedge_R)$	$\lceil A \wedge B \rceil^\perp \otimes (\lceil A \rceil \,\&\, \lceil B \rceil)$
$(\vee_L)$	$\lfloor A \vee B \rfloor^\perp \otimes (?\lfloor A \rfloor \,\&\, ?\lfloor B \rfloor)$	$(\vee_R)$	$\lceil A \vee B \rceil^\perp \otimes (\lceil A \rceil \oplus \lceil B \rceil)$
$(\forall_L)$	$\lfloor \forall B \rfloor^\perp \otimes ?\lfloor Bx \rfloor$	$(\forall_R)$	$\lceil \forall B \rceil^\perp \otimes \forall x \lceil Bx \rceil$
$(\exists_L)$	$\lfloor \exists B \rfloor^\perp \otimes \forall x ?\lfloor Bx \rfloor$	$(\exists_R)$	$\lceil \exists B \rceil^\perp \otimes \lceil Bx \rceil$
		(t_R)	$\lceil t \rceil^\perp \otimes \top$
(Id_1)	$\lfloor B \rfloor^\perp \otimes \lceil B \rceil^\perp$	(Id_2')	$?\lfloor B \rfloor \otimes !\lceil B \rceil$

Figure 4.6: The theory $\mathcal{L}_{lm}$ encodes the sequent calculus proof system LM^{c}.

$$(\perp_L) \quad \lfloor \perp \rfloor^\perp \qquad (W_R) \quad \lceil C \rceil^\perp \otimes \perp$$

Figure 4.7: Adding these two clauses to $\mathcal{L}_{lm}$ yields $\mathcal{L}_{lj}$, which is used to encode the sequent calculus proof system LJ^{c}.

$(\Rightarrow_L)$	$\lfloor A \Rightarrow B \rfloor^\perp \otimes (?\lceil A \rceil \otimes ?\lfloor B \rfloor)$	$(\Rightarrow_R)$	$\lceil A \Rightarrow B \rceil^\perp \otimes (?\lfloor A \rfloor \parr ?\lceil B \rceil)$
$(\wedge_L)$	$\lfloor A \wedge B \rfloor^\perp \otimes (?\lfloor A \rfloor \oplus ?\lfloor B \rfloor)$	$(\wedge_R)$	$\lceil A \wedge B \rceil^\perp \otimes (?\lceil A \rceil \,\&\, ?\lceil B \rceil)$
$(\vee_L)$	$\lfloor A \vee B \rfloor^\perp \otimes (?\lfloor A \rfloor \,\&\, ?\lfloor B \rfloor)$	$(\vee_R)$	$\lceil A \vee B \rceil^\perp \otimes (?\lceil A \rceil \oplus ?\lceil B \rceil)$
$(\forall_L)$	$\lfloor \forall B \rfloor^\perp \otimes ?\lfloor Bx \rfloor$	$(\forall_R)$	$\lceil \forall B \rceil^\perp \otimes \forall x ?\lceil Bx \rceil$
$(\exists_L)$	$\lfloor \exists B \rfloor^\perp \otimes \forall x ?\lfloor Bx \rfloor$	$(\exists_R)$	$\lceil \exists B \rceil^\perp \otimes ?\lceil Bx \rceil$
$(\perp_L)$	$\lfloor \perp \rfloor^\perp$	(t_R)	$\lceil t \rceil^\perp \otimes \top$
(Id_1)	$\lfloor B \rfloor^\perp \otimes \lceil B \rceil^\perp$	(Id_2)	$?\lfloor B \rfloor \otimes ?\lceil B \rceil$

Figure 4.8: The theory $\mathcal{L}_{lk}$ encodes the sequent calculus proof system LK^{c}.

4.3 Sequent Calculus

Figures 4.3, 4.4, and 4.5, respectively, contain three sequent calculi for minimal (LM^{c}), intuitionistic (LJ^{c}), and classical logic (LK^{c}), where contractions are implicit in the logical rules. A linear logic encoding for these systems is given by the theories, $\mathcal{L}_{lm}$, $\mathcal{L}_{lj}$ and $\mathcal{L}_{lk}$ shown in Figures 4.6, 4.7 and 4.8. These sets differ in the presence or absence of ? in front of $\lceil \cdot \rceil$, in the presence or absence of the formula $(\perp_L)$ and in the formula encoding the left introduction for implication. In particular, in the LM^{c} encoding, no structural rule is allowed for right-hand-side formulas; in the LJ^{c} encoding, the right-hand-side formulas can be weakened; and in the LK^{c} encoding, contraction is also allowed (using the exponential ?). The formula $(\perp_L)$ only appears in the encodings of LJ^{c} and LK^{c}. In the theories for LM^{c} and LJ^{c}, the formulas encoding the left introduction rule for implication and the formula Id_2' contain a ! before a positive occurrence of $\lceil \cdot \rceil$ atom. As we shall see, these occurrences of ! are necessary for preserving the invariant that in minimal and intuitionistic logics the right-hand-side of sequents do not contain more than one formula.

A key ingredient in capturing object-level sequent calculus inferences in a focused linear meta-logic is the assignment of negative polarity to all meta-level atomic formulas. To illustrate why focusing is relevant, consider the encoding of the left introduction rule for $\supset$: selecting this rule at the object-level corresponds to focusing on the formula $F = \exists A \exists B[\lfloor A \supset B \rfloor^\perp \otimes (!\lceil A \rceil \otimes \lfloor B \rfloor)]$ (which is a member of $\mathcal{L}_{lm}$). The focused derivation in Figure 4.9 is then forced once F is selected for the focus: for example, the left-hand-side subproof must be an application of initial – nothing else will work with the focusing discipline. Notice that this meta-level derivation directly encodes the usual left introduction rule for $\supset$: the object-level sequents $\Gamma, A \supset B, B \vdash C$ and $\Gamma, A \supset B \vdash A$ yields $\Gamma, A \supset B \vdash C$. Moreover, the ! enforces that in all branches there is at most one $\lceil \cdot \rceil$ atom.

If we fix the polarity of all meta-level atoms to be negative, then focused proofs using $\mathcal{L}_{lm}$, $\mathcal{L}_{lj}$, and

$$\dfrac{\dfrac{}{\vdash \mathcal{K} :\Downarrow \lfloor A \supset B \rfloor^{\perp}}\,[I_2] \quad \dfrac{\dfrac{\vdash \mathcal{K} : \lceil A \rceil \Uparrow}{\vdash \mathcal{K} :\Downarrow\, !\lceil A \rceil}\,[!, R\Uparrow] \quad \dfrac{\vdash \mathcal{K}, \lfloor B \rfloor : \lceil C \rceil \Uparrow}{\vdash \mathcal{K} : \lceil C \rceil \Downarrow\, ?\lfloor B \rfloor}\,[R\Downarrow, ?]}{\vdash \mathcal{K} : \lceil C \rceil \Downarrow\, !\lceil A \rceil \otimes\, ?\lfloor B \rfloor}\,[\otimes]}{\dfrac{\vdash \mathcal{K} : \lceil C \rceil \Downarrow F}{\vdash \mathcal{K} : \lceil C \rceil \Uparrow \cdot}\,[D_2]}\,[2 \times \exists, \otimes]$$

Figure 4.9: Here, the formula $A \supset B \in \Gamma$ and $\mathcal{K}$ denotes the set $\mathcal{L}_{lm}, \lfloor \Gamma \rfloor$.

$\mathcal{L}_{lk}$ yield encodings of the object-level proofs in LM^{c}, LJ^{c}, and LK^{c}. We use the judgments $\vdash_{lm}, \vdash_{lj}$, and $\vdash_{lk}$ to denote provability in $\mathrm{LM}^{\mathrm{c}}, \mathrm{LJ}^{\mathrm{c}}$, and LK^{c}.

Proposition 4.1 *Let $\Gamma \cup \Delta \cup \{C\}$ be a set of object-level formulas. Assume that all meta-level atomic formulas are given a negative polarity. Then*

$$\begin{array}{l} 1)\ \Gamma \vdash_{lm} C \text{ iff } \vdash_{llf} \mathcal{L}_{lm}, \lfloor \Gamma \rfloor : \lceil C \rceil \Uparrow \\ 2)\ \Gamma \vdash_{lj} C \text{ iff } \vdash_{llf} \mathcal{L}_{lj}, \lfloor \Gamma \rfloor : \lceil C \rceil \Uparrow \\ 3)\ \Gamma \vdash_{lk} \Delta \text{ iff } \vdash_{llf} \mathcal{L}_{lk}, \lfloor \Gamma \rfloor, \lceil \Delta \rceil : \cdot \Uparrow \end{array}$$

Furthermore, adequacy for derivations also holds between the respective proof systems.

Proof First, one shows that focusing (deciding) on formulas within the linear logic theories $\mathcal{L}_{lm}$, $\mathcal{L}_{lj}$, and $\mathcal{L}_{lk}$ encodes exactly the corresponding sequent calculus inference rule. In all cases, this correspondence is shown with steps similar to the one offered above for the left-introduction of $\supset$. Once this level of adequacy for the encoding is established, the other results concerning the equivalences of provability follow immediately. See also [Miller 2002, Pimentel 2001] for similar proofs related to the encoding of sequent calculus proofs. □

If one removes the formula Id_2 and Id_2' from the sets $\mathcal{L}_{lm}$, $\mathcal{L}_{lj}$, and $\mathcal{L}_{lk}$, obtaining the sets $\mathcal{L}^f_{lm}$, $\mathcal{L}^f_{lj}$, and $\mathcal{L}^f_{lk}$, respectively, one can restrict the encoded proofs to cut free (object-level) proofs, represented by the judgments $\vdash^f_{lm}$ for minimal logic, $\vdash^f_{lj}$ for intuitionistic logic, and $\vdash^f_{lk}$ for classical logic. The following proposition is an immediate consequence of the proof of Proposition 4.1.

Proposition 4.2 *Let $\Gamma \cup \Delta \cup \{C\}$ be a set of object-level formulas. Then*

$$\begin{array}{ll} 1)\ \Gamma \vdash^f_{lm} C \text{ iff } \vdash_{llf} \mathcal{L}^f_{lm}, \lfloor \Gamma \rfloor : \lceil C \rceil \Uparrow & 2)\ \Gamma \vdash^f_{lj} C \text{ iff } \vdash_{llf} \mathcal{L}^f_{lj}, \lfloor \Gamma \rfloor : \lceil C \rceil \Uparrow \\ 3)\ \Gamma \vdash^f_{lk} \Delta \text{ iff } \vdash_{llf} \mathcal{L}^f_{lk}, \lfloor \Gamma \rfloor, \lceil \Delta \rceil :\Uparrow & \end{array}$$

Furthermore, adequacy for derivations also holds between the respective proof systems.

Now that we have succeeded to find linear logic theories that encode the sequent calculus inference rules for minimal, intuitionistic, and classical logics at our strongest level of adequacy, we turn to showing how these theories are related back to the more elementary and modular sets of formulas shown in Figures 4.1 and 4.2. The equivalences that appear in the following three propositions are all at the most shallow level of adequacy: the equivalence of provability.

Proposition 4.3 *Let Γ and Δ be sets of object logic formulas. Then*

$$\vdash_{ll} \mathcal{L}, Id_1, Id_2, Str_L, Str_R, ?\lfloor \Gamma \rfloor, ?\lceil \Delta \rceil \text{ iff } \vdash_{ll} \mathcal{L}_{lk}, ?\lfloor \Gamma \rfloor, ?\lceil \Delta \rceil.$$

Proof From the structural rules, Str_L and Str_R, we know that $\lfloor C \rfloor \equiv ?\lfloor C \rfloor$ and $\lceil C \rceil \equiv ?\lceil C \rceil$. Since the only difference between $\mathcal{L}_{lk}$ and $\mathcal{L} \cup \{Id_1, Id_2\}$ is that the former has ? before positive occurrences of $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$, it is the case that $\mathcal{L}_{lk}$ is a consequence of $\mathcal{L} \cup \{Id_1, Id_2, Str_L, Str_R\}$, proving the $\Leftarrow$ direction.

For the $\Rightarrow$ direction, we need to show that the structural rules are admissible. We use focusing to help. In particular, we show that if $\vdash_{\mathit{llf}} \mathcal{L}, Id_1, Id_2, Str_L, Str_R, \mathcal{F}_1 : \mathcal{F}_2 \Uparrow$ then $\vdash_{\mathit{llf}} \mathcal{L}_{lk}, \mathcal{F}_1, \mathcal{F}_2 : \cdot \Uparrow$, where $\mathcal{F}_1$ and $\mathcal{F}_2$ are multisets of meta-level atoms (of which all are given a negative polarity). This is proved by induction on the height of focused proofs (the proof follows the same lines as in [Miller 2004, Proposition 4.2]). We show the inductive case for $(\Rightarrow_L)$: all the others cases are done similarly. Thus, assume that our proof ends with a decide rule that selects an instance of the $(\Rightarrow_L)$ formula from Figure 4.1. Thus, the proof ends with the following derivation, where $\mathcal{K} = \mathcal{L}, Id_1, Id_2, Str_L, Str_R, \mathcal{F}_1$ and $\mathcal{F}_2 = \mathcal{F}_2^1 \cup \mathcal{F}_2^2$ (here, $\mathcal{F}_1$ and $\mathcal{F}_2$ are multisets of atomic formulas).

$$\dfrac{\dfrac{\dfrac{}{\vdash \mathcal{K} : \cdot \Downarrow \lfloor A \Rightarrow B \rfloor^{\perp}}\,[I_2] \quad \dfrac{\vdash \mathcal{K} : \mathcal{F}_2^1, \lceil A \rceil \Uparrow}{\vdash \mathcal{K} : \mathcal{F}_2^1 \Downarrow \lceil A \rceil}\,[R\Downarrow, R\Uparrow] \quad \dfrac{\vdash \mathcal{K} : \mathcal{F}_2^2, \lfloor B \rfloor \Uparrow}{\vdash \mathcal{K} : \mathcal{F}_2^2 \Downarrow \lfloor B \rfloor}\,[R\Downarrow, R\Uparrow]}{\vdash \mathcal{K} : \cdot \Downarrow \lfloor A \Rightarrow B \rfloor^{\perp} \otimes (\lceil A \rceil \otimes \lfloor B \rfloor)}\,[2 \times \otimes]}{\vdash \mathcal{K} : \mathcal{F}_2 \Uparrow}\,[D_2, 2 \times \exists]$$

Thus, $\lfloor A \Rightarrow B \rfloor \in \mathcal{F}_1$, and by the induction hypothesis, we have proofs of the sequents $\vdash \mathcal{L}_{lk}, \mathcal{F}_1 : \mathcal{F}_2^1, \lceil A \rceil \Uparrow$ and $\vdash \mathcal{L}_{lk}, \mathcal{F}_1 : \mathcal{F}_2^2, \lfloor B \rfloor \Uparrow$. By Proposition 2.7, the sequents $\vdash \mathcal{K}', \lceil A \rceil : \cdot \Uparrow$ and $\vdash \mathcal{K}', \lfloor B \rfloor : \cdot \Uparrow$ are also provable, where $\mathcal{K}' = \mathcal{L}_{lk}, \mathcal{F}_1, \mathcal{F}_2$. Thus, the desired proof using the theory $\mathcal{L}_{lk}$ but with focusing on the $(\Rightarrow_L)$ formula in $\mathcal{L}_{lk}$ is

$$\dfrac{\dfrac{\dfrac{}{\vdash \mathcal{K}' : \cdot \Downarrow \lfloor A \Rightarrow B \rfloor^{\perp}}\,[I_2] \quad \dfrac{\vdash \mathcal{K}', \lceil A \rceil : \cdot \Uparrow}{\vdash \mathcal{K}' : \cdot \Downarrow ? \lceil A \rceil}\,[R\Downarrow, ?] \quad \dfrac{\vdash \mathcal{K}', \lfloor B \rfloor : \cdot \Uparrow}{\vdash \mathcal{K}' : \cdot \Downarrow ? \lfloor B \rfloor}\,[R\Downarrow, ?]}{\vdash \mathcal{K}' : \cdot \Downarrow \lfloor A \Rightarrow B \rfloor^{\perp} \otimes (? \lceil A \rceil \otimes ? \lfloor B \rfloor)}\,[2 \times \otimes]}{\vdash \mathcal{K}' : \cdot \Uparrow}\,[D_2, 2 \times \exists]$$

The $\Rightarrow$ direction is a direct consequence of this intermediate result and the focusing theorem. □

Proposition 4.4 *Let $\Gamma \cup \{C\}$ be a set of object logic formulas. Then*

1) $\vdash_{\mathit{ll}} \mathcal{L}_M, Id_1, Id_2', Str_L, ? \lfloor \Gamma \rfloor, \lceil C \rceil$ *iff* $\vdash_{\mathit{ll}} \mathcal{L}_{lm}, ? \lfloor \Gamma \rfloor, \lceil C \rceil$.
2) $\vdash_{\mathit{ll}} \mathcal{L}_J, Id_1, Id_2', Str_L, W_R, ? \lfloor \Gamma \rfloor, \lceil C \rceil$ *iff* $\vdash_{\mathit{ll}} \mathcal{L}_{lj}, ? \lfloor \Gamma \rfloor, \lceil C \rceil$.

Proof In the $\Rightarrow$ direction, we proceed in the same fashion as in Proposition 4.3. We prove that, for say minimal logic, if $\vdash_{\mathit{ll}} \mathcal{L}_M, Id_1, Id_2', Str_L, \mathcal{F}_1 : \mathcal{F}_2, \lceil C \rceil \Uparrow$ then $\vdash_{\mathit{ll}} \mathcal{L}_{lm}, \mathcal{F}_1, \mathcal{F}_2 : \lceil C \rceil \Uparrow$, where $\mathcal{F}_1 \cup \mathcal{F}_2$ is a multiset of $\lfloor \cdot \rfloor$ meta-level atoms and C is any object-logic formula. The main interesting case is when the proof of $\vdash \mathcal{K} : \mathcal{F}_2, \lceil C \rceil \Uparrow$ starts by focusing on $(\supset_L)$, where $\mathcal{K} = \mathcal{L}_M, Id_1, Id_2', Str_L, \mathcal{F}_1$. There is only one resulting focused derivation, due to the presence of the bang in $(\supset_L)$, and it has two open premises of the form $\vdash \mathcal{K} : \mathcal{F}_2^1, \lfloor B \rfloor, \lceil C \rceil \Uparrow$ and $\vdash \mathcal{K} : \mathcal{F}_2^2, \lceil A \rceil \Uparrow$, in which case the proof proceeds the same as in Proposition 4.3. □

Proposition 4.5 *Let $\Gamma \cup \Delta \cup \{C\}$ be a set of object logic formulas. Then*

1) $\vdash_{\mathit{ll}} \mathcal{L}_M, Id_1, Str_L, ? \lfloor \Gamma \rfloor, \lceil C \rceil$ *iff* $\vdash_{\mathit{ll}} \mathcal{L}_{lm}^f, ? \lfloor \Gamma \rfloor, \lceil C \rceil$
2) $\vdash_{\mathit{ll}} \mathcal{L}_J, Id_1, Str_L, W_R, ? \lfloor \Gamma \rfloor, \lceil C \rceil$ *iff* $\vdash_{\mathit{ll}} \mathcal{L}_{lj}^f, ? \lfloor \Gamma \rfloor, \lceil C \rceil$
3) $\vdash_{\mathit{ll}} \mathcal{L}, Id_1, Str_L, Str_R, ? \lfloor \Gamma \rfloor, ? \lceil \Delta \rceil$ *iff* $\vdash_{\mathit{ll}} \mathcal{L}_{lk}^f, ? \lfloor \Gamma \rfloor, ? \lceil \Delta \rceil$.

Proof This proposition is proved in a similar way as the Propositions 4.3 and 4.4. □

It is well known that for the sequent calculus systems LM^{c}, LJ^{c}, and LK^{c} the cut-elimination theorem holds. A direct consequence is the admissibility of the Id_2 rule in the theories considered for these sequent calculus systems, as states the following proposition.

Corollary 4.6 *Let $\Gamma \cup \Delta \cup \{C\}$ be a set of object logic formulas. Then*

1) $\vdash_{ll} \mathcal{L}_M, Id_1, Str_L, ?\lfloor\Gamma\rfloor, \lceil C\rceil$ *iff* $\vdash_{ll} \mathcal{L}_M, Id_1, Id'_2, Str_L, ?\lfloor\Gamma\rfloor, \lceil C\rceil$
2) $\vdash_{ll} \mathcal{L}_J, Id_1, Str_L, W_R, ?\lfloor\Gamma\rfloor, \lceil C\rceil$ *iff* $\vdash_{ll} \mathcal{L}_J, Id_1, Id'_2, Str_L, W_R, ?\lfloor\Gamma\rfloor, \lceil C\rceil$
3) $\vdash_{ll} \mathcal{L}, Id_1, Str_L, Str_R, ?\lfloor\Gamma\rfloor, ?\lceil\Delta\rceil$ *iff* $\vdash_{ll} \mathcal{L}, Id_1, Id_2, Str_L, Str_R, ?\lfloor\Gamma\rfloor, ?\lceil\Delta\rceil$.

The proof of this corollary follows from the admissibility of the cut rule [Gentzen 1969] and the encoding of the cut-free sequent calculus (Proposition 4.2). To see a setting in which the admissibility of the cut can be shown by directly considering the linear logic specification of inference rules, see [Miller 2002, Pimentel 2005].

4.4 Natural Deduction

The proof system depicted in Figure 4.10 is the $\forall$, $\wedge$, and $\supset$ intuitionistic fragment of the classical system in [Sieg 1998], presenting natural deduction using a sequent-style notation: sequents of the form $\Gamma \vdash C \uparrow$ are obtained from the conclusion by a derivation (reading bottom-up) where C is not the major premise of an elimination rule; and sequents of the form $\Gamma \vdash C \downarrow$ are obtained from the set of hypotheses by a derivation (from top-down) where C is extracted from the major premise of an elimination rule. These two types of derivations meet with either the match rule M or the switch rule S. These two types of sequents can be used to distinguish general natural deduction proofs from *normal* form proofs [Prawitz 1965]: normal proofs are those in which the major premise of an elimination rule is not the conclusion of an introduction rule. Within the proof system in Figure 4.10, such proofs are exactly those that do not allow occurrences of the switch rule S. To the rules in Figure 4.10 we can add the introduction and elimination rules for $\vee$ and $\exists$ given in Figure 4.11. In those rules, occurrences of $\uparrow(\downarrow)$ denote either $\uparrow$ or $\downarrow$ with the proviso that all occurrences of $\uparrow(\downarrow)$ in a given inference rule are resolved the same way. Characterizing normal form proofs involving $\vee$ and $\exists$ is more involved to describe and we shall not consider such normal forms here.

We write $\Gamma \vdash_{nj} C$ to indicate that the natural deduction sequent $\Gamma \vdash C \uparrow$ has a proof in NJ and write $\Gamma \vdash^{n}_{nj} C$ to indicate that the natural deduction sequent $\Gamma \vdash C \uparrow$ has a normal proof in NJ: in this latter case, we shall restrict the formulas in $\Gamma \cup \{C\}$ to have no occurrences of $\vee$ and $\exists$.

The theory $\mathcal{L}_{nj}$ in Figure 4.12 encodes natural deduction for intuitionistic logic. The formula Str_L is incorporated in the theory by adding ? to some positive occurrences of $\lfloor\cdot\rfloor$ atoms and to maintain the invariant that there is always at most one formula in the right-hand-side of sequents, we add ! to negative occurrences of $\lfloor\cdot\rfloor^{\perp}$. The judgment $\Gamma \vdash C \uparrow$ is encoded as the meta-level sequent $\vdash \mathcal{L}_{nj}, \lfloor\Gamma\rfloor : \lceil C\rceil$ and the judgment $\Gamma \vdash C \downarrow$ is encoded as the sequent $\vdash \mathcal{L}_{nj}, \lfloor\Gamma\rfloor : \lfloor C\rfloor^{\perp}$. In order for this encoding to be adequate at the level of derivations, we simply change the polarity assignment from what was used with sequent calculus: in particular, we assign atoms of the form $\lfloor\cdot\rfloor$ with positive polarity and atoms of the form $\lceil\cdot\rceil$ with negative polarity. This change in polarity changes left-introduction rules (within the sequent calculus) to elimination rules (within natural deduction). For example, the formula $(\supset_L)$ now encodes the implication elimination rule as is illustrated by the following derivation (here, $(\supset_L) \in \mathcal{K}$):

$$\dfrac{\dfrac{\dfrac{\vdash \mathcal{K} : \lfloor A \supset B\rfloor^{\perp} \Uparrow}{\vdash \mathcal{K} :\Downarrow \lfloor A \supset B\rfloor^{\perp}}\,[R\Downarrow, R\Uparrow] \quad \dfrac{\vdash \mathcal{K} : \lceil A\rceil \Uparrow}{\vdash \mathcal{K} :\Downarrow !\lceil A\rceil}\,[!, R\Uparrow] \quad \dfrac{}{\vdash \mathcal{K} : \lfloor B\rfloor^{\perp} \Downarrow \lfloor B\rfloor}\,[I_1]}{\vdash \mathcal{K} : \lfloor B\rfloor^{\perp} \Downarrow \lfloor A \supset B\rfloor^{\perp} \otimes (!\lceil A\rceil \otimes \lfloor B\rfloor)}\,[2\times\otimes]}{\vdash \mathcal{K} : \lfloor B\rfloor^{\perp} \Uparrow}\,[D_2, 2\times\exists]$$

The change in the assignment of polarity also causes the formula Id'_2, which behaved like the cut rule in sequent calculus, to now behave like the switch rule, as illustrated by the following derivation,

$$\frac{\Gamma \vdash A \supset B \downarrow \quad \Gamma \vdash A \uparrow}{\Gamma \vdash B \downarrow}\ [\supset E] \qquad \frac{\Gamma, A \vdash B \uparrow}{\Gamma \vdash A \supset B \uparrow}\ [\supset I]$$

$$\frac{\Gamma \vdash F \wedge G \downarrow}{\Gamma \vdash F \downarrow}\ [\wedge E] \qquad \frac{\Gamma \vdash F \uparrow \quad \Gamma \vdash G \uparrow}{\Gamma \vdash F \wedge G \uparrow}\ [\wedge I]$$

$$\frac{\Gamma \vdash \forall x\, A \downarrow}{\Gamma \vdash A\{t/x\} \downarrow}\ [\forall E] \qquad \frac{\Gamma \vdash A\{c/x\} \uparrow}{\Gamma \vdash \forall x\, A \uparrow}\ [\forall I]$$

$$\frac{}{\Gamma, A \vdash A \downarrow}\ [\mathrm{I}] \quad \frac{\Gamma \vdash A \downarrow}{\Gamma \vdash A \uparrow}\ [\mathrm{M}] \quad \frac{\Gamma \vdash A \uparrow}{\Gamma \vdash A \downarrow}\ [\mathrm{S}] \quad \frac{}{\Gamma \vdash t \uparrow}\ [tI] \quad \frac{\Gamma \vdash \bot \downarrow}{\Gamma \vdash C \uparrow}\ [\bot E]$$

Figure 4.10: The rules for the $\supset$, $\forall$, and $\wedge$ fragment of intuitionistic natural deduction NJ.

$$\frac{\Gamma \vdash A \vee B \downarrow \quad \Gamma, A \vdash C \uparrow(\downarrow)\ \Gamma, B \vdash C \uparrow(\downarrow)}{\Gamma \vdash C \uparrow(\downarrow)}\ [\vee E] \qquad \frac{\Gamma \vdash A_i \uparrow}{\Gamma \vdash A_1 \vee A_2 \uparrow}\ [\vee I]$$

$$\frac{\Gamma \vdash \exists x\, A \downarrow \quad \Gamma, A\{c/x\} \vdash C \uparrow(\downarrow)}{\Gamma \vdash C \uparrow(\downarrow)}\ [\exists E] \qquad \frac{\Gamma \vdash A\{t/x\} \uparrow}{\Gamma \vdash \exists x\, A \uparrow}\ [\exists I]$$

Figure 4.11: The rules for $\vee$ and $\exists$ for intuitionistic natural deduction. In $\vee L$, $i \in \{1, 2\}$.

$(\supset_E)$	$\lfloor A \supset B \rfloor^\perp \otimes (!\lceil A \rceil \otimes \lfloor B \rfloor)$	$(\supset_I)$	$\lceil A \supset B \rceil^\perp \otimes (?\lfloor A \rfloor \parr \lceil B \rceil)$
$(\wedge_E)$	$\lfloor A \wedge B \rfloor^\perp \otimes (\lfloor A \rfloor \oplus \lfloor B \rfloor)$	$(\wedge_I)$	$\lceil A \wedge B \rceil^\perp \otimes (\lceil A \rceil \,\&\, \lceil B \rceil)$
$(\vee_E)$	$!\lfloor A \vee B \rfloor^\perp \otimes (?\lfloor A \rfloor \,\&\, ?\lfloor B \rfloor)$	$(\vee_I)$	$\lceil A \vee B \rceil^\perp \otimes (\lceil A \rceil \oplus \lceil B \rceil)$
$(\forall_E)$	$\lfloor \forall B \rfloor^\perp \otimes \lfloor Bx \rfloor$	$(\forall_I)$	$\lceil \forall B \rceil^\perp \otimes \forall x \lceil Bx \rceil$
$(\exists_E)$	$!\lfloor \exists B \rfloor^\perp \otimes \forall x ?\lfloor Bx \rfloor$	$(\exists_I)$	$\lceil \exists B \rceil^\perp \otimes \lceil Bx \rceil$
$(\bot)$	$\lfloor \bot \rfloor^\perp$	(t_I)	$\lceil t \rceil^\perp \otimes \top$
$(\bot_E)$	$\lceil C \rceil^\perp \otimes \bot$		
(Id_1)	$\lfloor B \rfloor^\perp \otimes \lceil B \rceil^\perp$	(Id_2)	$\lfloor B \rfloor \otimes !\lceil B \rceil$

Figure 4.12: The specification $\mathcal{L}_{nj}$ for intuitionistic natural deduction.

where $Id'_2 \in \Sigma$.

$$\dfrac{\dfrac{\dfrac{}{\vdash \Sigma, \lfloor\Gamma\rfloor : \lfloor C\rfloor^{\perp} \Downarrow \lfloor C\rfloor}\,[I_1] \quad \dfrac{\vdash \Sigma, \lfloor\Gamma\rfloor : \lceil C\rceil \Uparrow}{\vdash \Sigma, \lfloor\Gamma\rfloor :\Downarrow\, !\lceil C\rceil}\,[!, R\Uparrow]}{\vdash \Sigma, \lfloor\Gamma\rfloor : \lfloor C\rfloor^{\perp} \Downarrow \lfloor C\rfloor \otimes\, !\lceil C\rceil}\,[\otimes]}{\vdash \Sigma, \lfloor\Gamma\rfloor : \lfloor C\rfloor^{\perp} \Uparrow}\,[D_2, \exists]$$

These two examples can be developed for all inference rules in Figures 4.10 and 4.11 and for focusing on all formulas in Figure 4.12. As the last example above suggests, we can capture normal natural deduction proofs if we remove instances of Id'_2 from $\mathcal{L}_{nj}$. More specifically, let $\mathcal{L}^{f}_{nj}$ be the set of formulas $\mathcal{L}_{nj}$ except that we drop Id'_2 and the formulas encoding the introduction rules for $\vee$ and $\exists$. As a result, it is an easy matter to prove the following proposition.

Proposition 4.7 *Let $\Gamma \cup \{C\}$ be a set of object-level formulas and assume that all $\lceil\cdot\rceil$ atomic formulas are given a negative polarity and that all $\lfloor\cdot\rfloor$ atomic formulas are given a positive polarity. Then $\Gamma \vdash_{nj} C$ if and only if $\vdash_{llf} \mathcal{L}_{nj}, \lfloor\Gamma\rfloor : \lceil C\rceil \Uparrow$. Also, if the formulas in $\Gamma \cup \{C\}$ contain neither $\vee$ nor $\exists$, then $\Gamma \vdash^{n}_{nj} C$ if and only if $\vdash_{llf} \mathcal{L}^{f}_{nj}, \lfloor\Gamma\rfloor : \lceil C\rceil \Uparrow$.*

Proof The proof is by structural induction over the height of trees. Most of the cases are included in the Appendix A.1 to further illustrate how these encodings work. □

Now that we have adequately encoded natural deduction derivations via the theory $\mathcal{L}_{nj}$, we can show how some (known) meta-theory results of intuitionistic logic can be achieved using these encodings. For example, we show in Proposition 4.10 below that sequent calculus proofs and natural deduction proofs prove the same formulas. First, the next two lemmas relate $\mathcal{L}_{nj}$ and $\mathcal{L}^{f}_{nj}$ with the formulas in Figure 4.1 and 4.2.

Lemma 4.8 *Let $\Gamma \cup \{C\}$ be a set of object logic formulas. Then*

$$\vdash_{ll} \mathcal{L}_J, Id_1, Id'_2, Str_L, W_R, ?\lfloor\Gamma\rfloor, \lceil C\rceil \text{ iff } \vdash_{ll} \mathcal{L}_{nj}, ?\lfloor\Gamma\rfloor, \lceil C\rceil.$$

Proof The proof follows the same lines as the proof of the Proposition 4.3. The main difference in the $\Leftarrow$ direction is that we also use the equivalence $\lfloor C\rfloor^{\perp} \equiv\, !\lfloor C\rfloor^{\perp}$ obtained from Str_L.

In the $\Rightarrow$ direction, we first prove the following equivalence, by induction on the height of proofs and by assigning negative polarity to all $\lceil\cdot\rceil$ atoms and positive polarity to all $\lfloor\cdot\rfloor$ atoms:

$$\vdash_{llf} \mathcal{L}_{nj}, \lfloor\Gamma\rfloor : \lceil C\rceil \Uparrow \text{ iff } \vdash_{llf} \mathcal{L}_{nj}, Str_L, \lfloor\Gamma\rfloor : \lceil C\rceil \Uparrow$$

The case for when Str_L is focused on is the most interesting one. There are two cases, either (1) the resulting premises are of the form $\vdash \mathcal{L}_{nj}, Str_L, \lfloor\Gamma\rfloor : \lfloor B\rfloor^{\perp} \Uparrow$ and $\vdash \mathcal{L}_{nj}, Str_L, \lfloor\Gamma, B\rfloor : \lceil C\rceil \Uparrow$, for which case we can use a linear-logic cut rule with cut formula $?\lfloor B\rfloor$: one premise is provable due to the induction hypothesis, and the other is provable also by the induction hypothesis, but by first introducing the ! in the cut formula $!\lfloor B\rfloor^{\perp}$; or (2) the premises are of the form $\vdash \mathcal{L}_{nj}, Str_L, \lfloor\Gamma\rfloor : \lfloor B\rfloor^{\perp}, \lceil C\rceil \Uparrow$ and $\vdash \mathcal{L}_{nj}, Str_L, \lfloor\Gamma, B\rfloor : \cdot \Uparrow$. In this case, because the elimination rules permute over introduction rules in natural deduction, we can assume that the proof of $\vdash \mathcal{L}_{nj}, Str_L, \lfloor\Gamma\rfloor : \lfloor B\rfloor^{\perp}, \lceil C\rceil \Uparrow$ finishes with a derivation that focuses only on formulas encoding (natural deduction) introduction rules and has premises of the form $\vdash \mathcal{L}_{nj}, Str_L, \lfloor\Gamma'\rfloor : \lfloor B\rfloor^{\perp} \Uparrow$. Here, there must be no other linear formula in the context, otherwise this sequent is not provable by applying only the encodings of (natural deduction) elimination rules, as these derivations would always contain a premise with at least two linear formulas, and hence one is never able to apply the initial rule. We then proceed as in the first case, but with the difference that we postpone the introduction of the bang of the cut formula, $!\lfloor B\rfloor^{\perp}$, until when these premises are reached.

From the Str_L formula we derive the equivalence $\lfloor C\rfloor^{\perp} \equiv\, !\lfloor C\rfloor^{\perp}$, which allows us to obtain the equivalent theory, $\mathcal{L}'_{nj}$, from $\mathcal{L}_{nj}$ by replacing all occurrences of $!\lfloor C\rfloor^{\perp}$ by $\lfloor C\rfloor^{\perp}$. Now, we show

the following intermediate result by induction on the height of proofs and using the same polarity assignment as before:

$$\vdash_{llf} \mathcal{L}'_{nj}, Str_L, \mathcal{F}_1, \mathcal{F}_2 : \lceil C \rceil \Uparrow \text{ iff } \vdash_{llf} \mathcal{L}_J, Id_1, Id'_2, Str_L, \mathcal{F}_1 : \mathcal{F}_2, \lceil C \rceil \Uparrow$$

where $\mathcal{F}_1$ and $\mathcal{F}_2$ are sets of $\lfloor \cdot \rfloor$ atoms and C an object-logic formula. This direction follows immediately from this intermediate result and the focusing theorem. □

The proof of the following lemma is similar to the proof of Lemma 4.8.

Lemma 4.9 *Let $\Gamma \cup \{C\}$ be a set of object logic formulas that do not contain occurrences of $\vee$ and $\exists$. Then*

$$\vdash_{ll} \mathcal{L}_J, Id_1, Str_L, W_R, ?\lfloor \Gamma \rfloor, \lceil C \rceil \text{ iff } \vdash_{ll} \mathcal{L}^f_{nj}, ?\lfloor \Gamma \rfloor, \lceil C \rceil.$$

From Propositions 4.4 and 4.5, Lemmas 4.8 and 4.9, and Propositions 4.1, 4.2, and 4.7, we obtain the following relative completeness result between LJ^c and NJ.

Proposition 4.10 *If $\Gamma \cup \{C\}$ be a set of object-level formulas, then $\Gamma \vdash_{lj} C$ if and only if $\Gamma \vdash_{nj} C$. Furthermore, if the formulas in $\Gamma \cup \{C\}$ contain neither $\vee$ nor $\exists$ then $\Gamma \vdash^f_{lj} C$ if and only if $\Gamma \vdash^n_{nj} C$.*

Treating negation (in particular, falsity) in natural deduction presentations of intuitionistic and classical logics is not straightforward. We show in [Nigam 2008c] that extra meta-logic formulas are needed to encode these systems. Since the treatment of negation in natural deduction is not one about focusing in the meta-level, we do not discuss this issue further here.

4.5 Natural Deduction with Generalized Elimination Rules

Schroeder-Heister [Schroeder-Heister 1984] considered a form of natural deduction where the indirect style of elimination rules used for $\vee$ and $\exists$ (see Figure 4.11) were also applied to conjunction. Von Plato [von Plato 2001] used that style of elimination rule for all connectives. In Figure 4.13 we present an additive version of a natural deduction system with generalized elimination inspired by one found in [Negri 2001, page 167]. The bracketed formula in an elimination rule is called the *major premise*. To encode proofs in natural deduction using generalized elimination, we use the theory $\mathcal{L}_{ge}$ shown in Figure 4.14. Intuitively, $\mathcal{L}_{ge}$ is obtained from $\mathcal{L}$ by using the formula Str_L to insert ! and ? connectives and using the identity rules to replace negative literals $\lfloor C \rfloor^\perp$ by the positive atoms $\lceil C \rceil$.

In order to match focused proofs using $\mathcal{L}_{ge}$ with the proofs in Figure 4.13, we assign negative polarity to all $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ meta-level atomic formulas. For example, focusing on the formula $(\supset_E)$ in Figure 4.14 yields the following derivation, where $\mathcal{K} = \mathcal{L}_{ge} \cup \lfloor \Gamma \rfloor$:

$$\dfrac{\dfrac{\dfrac{\vdash \mathcal{K} : \lceil A \supset B \rceil \Uparrow}{\vdash \mathcal{K} :\Downarrow !\lceil A \supset B \rceil}\,[!, R\Uparrow] \quad \dfrac{\vdash \mathcal{K} : \lceil A \rceil \Uparrow}{\vdash \mathcal{K} :\Downarrow !\lceil A \rceil}\,[!, R\Uparrow] \quad \dfrac{\vdash \mathcal{K}, \lfloor B \rfloor : \lceil C \rceil \Uparrow}{\vdash \mathcal{K} : \lceil C \rceil \Downarrow ?\lfloor B \rfloor}\,[R\Downarrow, ?]}{\vdash \mathcal{K} : \lceil C \rceil \Downarrow !\lceil A \supset B \rceil \otimes (!\lceil A \rceil \otimes ?\lfloor B \rfloor)}\,[2 \times \otimes]}{\vdash \mathcal{K} : \lceil C \rceil \Uparrow}\,[D_2, 2 \times \exists]$$

We can repeat this computation for all formulas in $\mathcal{L}_{ge}$ and in the process, prove the following proposition.

Proposition 4.11 *Let $\Gamma \cup \{C\}$ be a set of object-level formulas and assume that all meta-level atomic formulas are given a negative polarity. The sequent $\Gamma \vdash C$ is provable in GE if and only if $\vdash \mathcal{L}_{ge}, \lfloor \Gamma \rfloor : \lceil C \rceil \Uparrow$ is provable in LLF. Furthermore, adequacy for derivations also holds between the respective proof systems.*

$$\frac{\Gamma \vdash [A \supset B] \quad \Gamma \vdash A \quad \Gamma, B \vdash C}{\Gamma \vdash C}\ [\supset GE] \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B}\ [\supset I]$$

$$\frac{\Gamma \vdash [A \wedge B] \quad \Gamma, A, B \vdash C}{\Gamma \vdash C}\ [\wedge GE] \qquad \frac{\Gamma \vdash F \quad \Gamma \vdash G}{\Gamma \vdash F \wedge G}\ [\wedge I]$$

$$\frac{\Gamma \vdash [A \vee B] \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}\ [\vee GE] \qquad \frac{\Gamma \vdash A_i}{\Gamma \vdash A_1 \vee A_2}\ [\vee I]$$

$$\frac{\Gamma \vdash [\forall x\, A] \quad \Gamma, A\{t/x\} \vdash C}{\Gamma \vdash C}\ [\forall GE] \qquad \frac{\Gamma \vdash A\{c/x\}}{\Gamma \vdash \forall x\, A}\ [\forall I]$$

$$\frac{\Gamma \vdash [\exists x\, A] \quad \Gamma, A\{c/x\} \vdash C}{\Gamma \vdash C}\ [\exists GE] \qquad \frac{\Gamma \vdash A\{t/x\}}{\Gamma \vdash \exists x\, A}\ [\exists I]$$

$$\frac{}{\Gamma, A \vdash A}\ [\mathrm{I}] \qquad \frac{}{\Gamma \vdash t}\ [tI] \qquad \frac{\Gamma \vdash \bot}{\Gamma \vdash C}\ [\bot E]$$

Figure 4.13: The rules for intuitionistic natural deduction system with generalized elimination rules, GE. The major premises of elimination rules is marked with brackets.

$(\supset_E)$	$!\lceil A \supset B\rceil \otimes (!\lceil A\rceil \otimes ?\lfloor B\rfloor)$	$(\supset_I)$	$\lceil A \supset B\rceil^{\perp} \otimes (?\lfloor A\rfloor \parr \lceil B\rceil)$
$(\wedge_E)$	$!\lceil A \wedge B\rceil \otimes (?\lfloor A\rfloor \parr ?\lfloor B\rfloor)$	$(\wedge_I)$	$\lceil A \wedge B\rceil^{\perp} \otimes (\lceil A\rceil \,\&\, \lceil B\rceil)$
$(\vee_E)$	$!\lceil A \vee B\rceil \otimes (?\lfloor A\rfloor \,\&\, ?\lfloor B\rfloor)$	$(\vee_I)$	$\lceil A \vee B\rceil^{\perp} \otimes (\lceil A\rceil \oplus \lceil B\rceil)$
$(\forall_E)$	$!\lceil \forall B\rceil \otimes ?\lfloor Bx\rfloor$	$(\forall_I)$	$\lceil \forall B\rceil^{\perp} \otimes \forall x \lceil Bx\rceil$
$(\exists_E)$	$!\lfloor \exists B\rfloor^{\perp} \otimes \forall x ?\lfloor Bx\rfloor$	$(\exists_I)$	$\lceil \exists B\rceil^{\perp} \otimes \lceil Bx\rceil$
$(\bot)$	$\lceil \bot\rceil$	(t_I)	$\lceil t\rceil^{\perp} \otimes \top$
$(\bot_E)$	$\lceil C\rceil^{\perp} \otimes \bot$		
(Id_1)	$\lfloor B\rfloor^{\perp} \otimes \lceil B\rceil^{\perp}$		

Figure 4.14: The specification $\mathcal{L}_{ge}$ for intuitionistic natural deduction with generalized elimination rules.

Proof The proof is by structural induction over the height of derivations. We show here only some of the cases, involving general elimination rules. The introduction rules are similar to the cases with introduction rules in the encoding of Natural Deduction in Proposition 4.7. In the derivations below $\mathcal{K} = \mathcal{L}_{ge} \cup \lfloor \Gamma \rfloor$.

$$\frac{\Gamma \vdash [A \wedge B] \quad \Gamma, A, B \vdash C}{\Gamma \vdash C}\,[\wedge GE] \quad \rightsquigarrow$$

$$\dfrac{\dfrac{\dfrac{\vdash \mathcal{K} : \lceil A \wedge B \rceil \Uparrow}{\vdash \mathcal{K} :\Downarrow\, !\lceil A \wedge B \rceil}\,[!, R\Uparrow] \quad \dfrac{\dfrac{\vdash \mathcal{K}, \lfloor A \rfloor, \lfloor B \rfloor : \lceil C \rceil \Uparrow}{\vdash \mathcal{K} : \lceil C \rceil \Uparrow ?\lfloor A \rfloor, ?\lfloor B \rfloor}\,[2 \times ?]}{\vdash \mathcal{K} : \lceil C \rceil \Downarrow ?\lfloor A \rfloor \parr ?\lfloor B \rfloor}\,[R\Uparrow, \parr]}{\vdash \mathcal{K} : \lceil C \rceil \Downarrow\, !\lceil A \wedge B \rceil \otimes (?\lfloor A \rfloor \parr ?\lfloor B \rfloor)}\,[\otimes]}{\vdash \mathcal{K} : \lceil C \rceil \Uparrow}\,[D_2, 2 \times \exists]$$

$$\frac{\Gamma \vdash [\forall x\, A] \quad \Gamma, A\{t/x\} \vdash C}{\Gamma \vdash C}\,[\forall GE] \quad \rightsquigarrow$$

$$\dfrac{\dfrac{\dfrac{\vdash \mathcal{K} : \lceil \forall B \rceil \Uparrow}{\vdash \mathcal{K} :\Downarrow\, !\lceil \forall B \rceil}\,[!, R\Uparrow] \quad \dfrac{\vdash \mathcal{K}, \lfloor Bt \rfloor : \lceil C \rceil \Uparrow}{\vdash \mathcal{K} : \lceil C \rceil \Downarrow ?\lfloor Bt \rfloor}\,[R\Uparrow, ?]}{\vdash \mathcal{K} : \lceil C \rceil \Downarrow\, !\lceil \forall B \rceil \otimes ?\lfloor Bt \rfloor}\,[\otimes]}{\vdash \mathcal{K} : \lceil C \rceil \Uparrow}\,[D_2, 2 \times \exists]$$

□

Given this linear logic theory, which encodes natural deduction with generalized elimination rules at our strongest level of adequacy, we turn to showing how $\mathcal{L}_{ge}$ relates back to the sets of formulas shown in Figures 4.1 and 4.2.

Proposition 4.12 *Let $\Gamma \cup \{C\}$ be a set of object logic formulas. Then, if $\vdash_{ll} \mathcal{L}_{ge}, ?\lfloor \Gamma \rfloor, \lceil C \rceil$ then $\vdash_{ll} \mathcal{L}_J, Id_1, Id_2', Str_L, ?\lfloor \Gamma \rfloor, \lceil C \rceil$. Furthermore, if $\vdash_{ll} \mathcal{L}_J, Id_1, Str_L, ?\lfloor \Gamma \rfloor, \lceil C \rceil$ then $\vdash_{ll} \mathcal{L}_{ge}, ?\lfloor \Gamma \rfloor, \lceil C \rceil$.*

Proof The second statement is proved in the same lines as in the proof of Proposition 4.3. For the first statement, we use a theory $\mathcal{L}_J'$, equivalent to $\mathcal{L}_J$, that is obtained by replacing literals of the form $\lfloor C \rfloor^\perp$ by the formula $\lfloor C \rfloor^\perp \parr \perp$, in the clauses $(\vee_L), (\wedge_L), (\supset_L)$, and $(\forall_L)$ in $\mathcal{L}_J$. Although $\lfloor C \rfloor^\perp$ and $\lfloor C \rfloor^\perp \parr \perp$ are logically equivalent, they have different focusing behaviors, as the latter has negative polarity regardless of the polarity given to $\lfloor C \rfloor$. Now, we assign negative polarity to all meta-level atoms and prove, by induction on the height of proofs, that if $\vdash_{llf} \mathcal{L}_{ge}, \mathcal{F}_1, \mathcal{F}_2 : \lceil C \rceil \Uparrow$ then $\vdash_{llf} \mathcal{L}_J', Id_1, Id_2', Str_L, \mathcal{F}_1 : \mathcal{F}_2, \lceil C \rceil \Uparrow$, where $\mathcal{F}_1 \cup \mathcal{F}_2$ is a multiset of $\lfloor \cdot \rfloor$ meta-level atoms. In this proof, when necessary, we use the formulas Id_2' and Str_L in $\mathcal{L}_J'$ to obtain a derivation for a sequent of the form $\vdash \mathcal{L}_J', Id_1, Id_2', Str_L, \mathcal{F}_1 : \mathcal{F}_2, \lfloor C \rfloor^\perp \Uparrow$ with open premise of the form $\vdash \mathcal{L}_J', Id_1, Id_2', Str_L, \mathcal{F}_1, \mathcal{F}_2 : \lceil C \rceil \Uparrow$. The statement follows directly from this intermediate result and the focusing theorem. □

Notice that from the lemma above, $\mathcal{L}_{ge}$'s expressiveness lies between a theory that does not contain Id_2' and that theory with Id_2'. From Corollary 4.6, however, we know that the Id_2' clause is admissible, so the following corollary holds.

Corollary 4.13 *Let $\Gamma \cup \{C\}$ be a set of object logic formulas. Then*

$$\vdash_{ll} \mathcal{L}_J, Id_1, Id_2', Str_L, ?\lfloor \Gamma \rfloor, \lceil C \rceil \text{ iff } \vdash_{ll} \mathcal{L}_{ge}, ?\lfloor \Gamma \rfloor, \lceil C \rceil$$

Although we obtain a theory that encodes GE with the strongest level of adequacy, we find it odd that $\mathcal{L}_{ge}$ does not relate so easily with other intuitionistic/minimal theories, since we used cut-elimination in the object-logic to establish the formal connection. We believe that the system as it is written does not pinpoint exactly where the clause Id_2' is needed. A similar problem happens in

$$\frac{\Gamma \vdash A \supset B \downarrow \quad \Gamma \vdash A \quad \Gamma, B \vdash C \uparrow(\downarrow)}{\Gamma \vdash C \uparrow(\downarrow)} [\supset GE] \qquad \frac{\Gamma, A \vdash B \uparrow}{\Gamma \vdash A \supset B \uparrow} [\supset I]$$

$$\frac{\Gamma \vdash A \wedge B \downarrow \quad \Gamma, A, B \vdash C \uparrow(\downarrow)}{\Gamma \vdash C \uparrow(\downarrow)} [\wedge GE] \qquad \frac{\Gamma \vdash F \uparrow \quad \Gamma \vdash G \uparrow}{\Gamma \vdash F \wedge G \uparrow} [\wedge I]$$

$$\frac{\Gamma \vdash A \vee B \downarrow \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C \uparrow(\downarrow)}{\Gamma \vdash C \uparrow(\downarrow)} [\vee GE] \qquad \frac{\Gamma \vdash A_i \uparrow}{\Gamma \vdash A_1 \vee A_2 \uparrow} [\vee I]$$

$$\frac{\Gamma \vdash \forall x\, A \downarrow \quad \Gamma, A\{t/x\} \vdash C \uparrow(\downarrow)}{\Gamma \vdash C \uparrow(\downarrow)} [\forall GE] \qquad \frac{\Gamma \vdash A\{c/x\} \uparrow}{\Gamma \vdash \forall x\, A \uparrow} [\forall I]$$

$$\frac{\Gamma \vdash \exists x\, A \downarrow \quad \Gamma, A\{c/x\} \vdash C \uparrow(\downarrow)}{\Gamma \vdash C \uparrow(\downarrow)} [\exists GE] \qquad \frac{\Gamma \vdash A\{t/x\} \uparrow}{\Gamma \vdash \exists x\, A \uparrow} [\exists I]$$

$$\frac{}{\Gamma, A \vdash A \downarrow} \text{[I]} \qquad \frac{\Gamma \vdash A \downarrow}{\Gamma \vdash A \uparrow} \text{[M]} \qquad \frac{\Gamma \vdash A \uparrow}{\Gamma \vdash A \downarrow} \text{[S]} \qquad \frac{}{\Gamma \vdash t \uparrow} [tI] \qquad \frac{\Gamma \vdash \bot \downarrow}{\Gamma \vdash C \uparrow} [\bot E]$$

Figure 4.15: The rules for the natural deduction with generalized elimination rules and with annotated sequents, GEA.

$(\supset_E)$	$!\lfloor A \supset B \rfloor^\perp \otimes (!\lceil A \rceil \otimes ?\lfloor B \rfloor)$	$(\supset_I)$	$\lceil A \supset B \rceil^\perp \otimes (?\lfloor A \rfloor \parr \lceil B \rceil)$
$(\wedge_E)$	$!\lfloor A \wedge B \rfloor^\perp \otimes (?\lfloor A \rfloor \parr ?\lfloor B \rfloor)$	$(\wedge_I)$	$\lceil A \wedge B \rceil^\perp \otimes (\lceil A \rceil \,\&\, \lceil B \rceil)$
$(\vee_E)$	$!\lfloor A \vee B \rfloor^\perp \otimes (?\lfloor A \rfloor \,\&\, ?\lfloor B \rfloor)$	$(\vee_I)$	$\lceil A \vee B \rceil^\perp \otimes (\lceil A \rceil \oplus \lceil B \rceil)$
$(\forall_E)$	$!\lfloor \forall B \rfloor^\perp \otimes ?\lfloor Bx \rfloor$	$(\forall_I)$	$\lceil \forall B \rceil^\perp \otimes \forall x \lceil Bx \rceil$
$(\exists_E)$	$!\lfloor \exists B \rfloor^\perp \otimes \forall x ?\lfloor Bx \rfloor$	$(\exists_I)$	$\lceil \exists B \rceil^\perp \otimes \lceil Bx \rceil$
$(\bot)$	$\lfloor \bot \rfloor^\perp$	(t_I)	$\lceil t \rceil^\perp \otimes \top$
$(\bot_E)$	$\lceil C \rceil^\perp \otimes \bot$		
(Id_1)	$\lfloor B \rfloor^\perp \otimes \lceil B \rceil^\perp$	(Id_2)	$\lfloor B \rfloor \otimes !\lceil B \rceil$

Figure 4.16: The specification $\mathcal{L}_{gea}$ for intuitionistic natural deduction with generalized elimination rules.

traditional presentations of natural deductions that do not use annotated sequents and do not contain the M and S rules (Figure 4.10). The S rule allows a natural deduction proof to have the major premise of an elimination rule be the conclusion of an introduction rule. Negri and von Plato in [Negri 2001] call such pairs of inference rules *detour cuts* and it is these pairs that correspond to the cut rule in sequent calculus. We present a variant of GE, called GEA (Figure 4.15), that makes these detour cuts apparent by using two types of annotated sequents: $\Gamma \vdash C \uparrow$ and $\Gamma \vdash C \downarrow$. We denote by the judgment $\vdash_{gea}$ provability in GEA (possibly containing the inference rule S and, hence, detour cuts), and we denote by the judgment $\vdash^{d}_{gea}$, provability from GEA without the inference rule S.

To encode GEA, we use the theory, $\mathcal{L}_{gea}$, shown in Figure 4.16, and we assign negative polarity to all $\lceil\cdot\rceil$ meta-level atoms and positive polarity to all $\lfloor\cdot\rfloor$ meta-level atoms. As before with natural deduction, the sequents $\Gamma \vdash C \uparrow$ and $\Gamma \vdash C \downarrow$ are encoded by meta-level sequents of the form $\vdash \mathcal{L}_{gea}, \lfloor\Gamma\rfloor : \lceil C\rceil \Uparrow$ and $\vdash \mathcal{L}_{gea}, \lfloor\Gamma\rfloor : \lfloor C\rfloor^{\perp} \Uparrow$, respectively. Now, the formula $(\supset_E)$ in $\mathcal{L}_{gea}$ encodes the generalized elimination rule for implication in GEA, as illustrated by the following derivation, where $\mathcal{K} = \mathcal{L}_{gea} \cup \lfloor\Gamma\rfloor$ and F is either $\lceil C\rceil$ or $\lfloor C\rfloor^{\perp}$:

$$\dfrac{\dfrac{\dfrac{\vdash \mathcal{K} : \lfloor A \supset B\rfloor^{\perp} \Uparrow}{\vdash \mathcal{K} : \cdot \Downarrow\, !\lfloor A \supset B\rfloor^{\perp}}\,[!, R\Uparrow] \quad \dfrac{\vdash \mathcal{K} : \lceil A\rceil \Uparrow}{\vdash \mathcal{K} : \cdot \Downarrow\, !\lceil A\rceil}\,[!, R\Uparrow] \quad \dfrac{\vdash \mathcal{K}, \lfloor B\rfloor : F \Uparrow}{\vdash \mathcal{K} : F \Downarrow\, ?\lfloor B\rfloor}\,[R\Downarrow, ?]}{\vdash \mathcal{K} : F \Downarrow\, !\lfloor A \supset B\rfloor^{\perp} \otimes (!\lceil A\rceil \otimes ?\lfloor B\rfloor)}\,[2\times\otimes]}{\vdash \mathcal{K} : F \Uparrow}\,[D_2, 2\times\exists]$$

We can repeat this style computation of focused derivation for every formula of $\mathcal{L}^{d}_{gea}$, thereby proving the following proposition.

Proposition 4.14 *Let $\Gamma \cup \{C\}$ be a set of object-level formulas and let $\mathcal{L}^{d}_{gea} = \mathcal{L}_{gea} \setminus \{Id_2\}$. Assume that all $\lceil\cdot\rceil$ atomic formulas are given a negative polarity and that all $\lfloor\cdot\rfloor$ atomic formulas are given a positive polarity. Then*

1) $\Gamma \vdash_{gea} C\uparrow$ *iff* $\vdash_{llf} \mathcal{L}_{gea}, \lfloor\Gamma\rfloor : \lceil C\rceil \Uparrow$ 2) $\Gamma \vdash^{d}_{gea} C\uparrow$ *iff* $\vdash_{llf} \mathcal{L}^{d}_{gea}, \lfloor\Gamma\rfloor : \lceil C\rceil \Uparrow$
3) $\Gamma \vdash^{d}_{gea} C\downarrow$ *iff* $\vdash_{llf} \mathcal{L}^{d}_{gea}, \lfloor\Gamma\rfloor : \lfloor C\rfloor^{\perp} \Uparrow$.

Proof The proof is similar to the proofs of Propositions 4.7 and 4.11. □

The following proposition can be proved similarly to the proof of the Lemma 4.8. This proposition provides the more careful placement of the Id'_2 meta-level axiom that motivated our introduction of the annotated proof system.

Proposition 4.15 *Let $\Gamma \cup \{C\}$ be a set of object logic formulas and let $\mathcal{L}^{d}_{gea} = \mathcal{L}_{gea} \setminus \{Id_2\}$. Then*

1) $\vdash_{ll} \mathcal{L}_J, Id_1, Str_L, W_R, ?\lfloor\Gamma\rfloor, \lceil C\rceil$ *iff* $\vdash_{ll} \mathcal{L}^{d}_{gea}, ?\lfloor\Gamma\rfloor, \lceil C\rceil$
2) $\vdash_{ll} \mathcal{L}_J, Id_1, Id'_2, Str_L, W_R, ?\lfloor\Gamma\rfloor, ?\lceil C\rceil$ *iff* $\vdash_{ll} \mathcal{L}_{gea}, ?\lfloor\Gamma\rfloor, \lceil C\rceil$.

Negri and von Plato in [Negri 2001] identify another type of cut, called *permutation cuts*, which occurs whenever the major premise of an elimination rule is the conclusion of another elimination rule. They also propose a different notion of normal proofs, called *general normal form*, for proofs in natural deduction with generalized elimination rules where both detour and permutation cuts do not appear. In particular, derivations in general normal form are such that the major premise of elimination rules are assumptions. In other words, the major premises in the generalized elimination rules shown in Figure 4.15, are discharged assumptions. We write $\Gamma \vdash^{n} C$ to denote that there is a general normal form proof of C from assumptions Γ. In our framework, this amounts to enforcing, by the use of polarity assignment to meta-level atoms, that the major premises are present in the set of assumptions. We use the theory $\mathcal{L}^{n}_{ge}$ obtained from $\mathcal{L}^{d}_{gea}$, by replacing formulas of the form $!\lfloor C\rfloor^{\perp}$ by $\lfloor C\rfloor^{\perp}$, and assign negative polarity to all atoms of the form $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$, to encode general normal form proofs, represented by the judgment $\vdash^{n}$.

$$\frac{\Gamma \vdash \Delta, A \Rightarrow B \quad \Gamma \vdash \Delta, A \quad \Gamma, B \vdash \Delta}{\Gamma \vdash \Delta}\ [\Rightarrow GE]$$

$$\frac{\Gamma, A \Rightarrow B \vdash \Delta \quad \Gamma, A \vdash \Delta}{\Gamma \vdash \Delta}\ [\Rightarrow GI_1] \qquad \frac{\Gamma, A \Rightarrow B \vdash \Delta \quad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta}\ [\Rightarrow GI_2]$$

$$\frac{\Gamma \vdash \Delta, A_1 \wedge A_2 \quad \Gamma, A_i \vdash \Delta}{\Gamma \vdash \Delta}\ [\wedge GE_i] \qquad \frac{\Gamma, A \wedge B \vdash \Delta \quad \Gamma \vdash \Delta, A \quad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta}\ [\wedge GI]$$

$$\frac{\Gamma \vdash \Delta, A \vee B \quad \Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma \vdash \Delta}\ [\vee GE] \qquad \frac{\Gamma, A_1 \vee A_2 \vdash \Delta \quad \Gamma \vdash \Delta, A_i}{\Gamma \vdash \Delta}\ [\vee GI_i]$$

$$\frac{}{\Gamma, A \vdash \Delta, A}\ [\mathrm{I}] \qquad \frac{\Gamma, \neg A \vdash \Delta \quad \Gamma, A \vdash \Delta}{\Gamma \vdash \Delta}\ [\neg GI_1] \qquad \frac{\Gamma \vdash \Delta, \neg A \quad \Gamma \vdash \Delta, A}{\Gamma \vdash \Delta}\ [\neg GI_2]$$

Figure 4.17: The rules for free deduction, FD.

$(\Rightarrow_E)$	$?\lceil A \Rightarrow B\rceil \otimes (?\lceil A\rceil \otimes ?\lfloor B\rfloor)$	$(\Rightarrow_I)$	$?\lfloor A \Rightarrow B\rfloor \otimes (?\lfloor A\rfloor \oplus ?\lceil B\rceil)$
$(\wedge_E)$	$?\lceil A \wedge B\rceil \otimes (?\lfloor A\rfloor \oplus ?\lfloor B\rfloor)$	$(\wedge_I)$	$?\lfloor A \wedge B\rfloor \otimes (?\lceil A\rceil \,\&\, ?\lceil B\rceil)$
$(\vee_E)$	$?\lceil A \vee B\rceil \otimes (?\lfloor A\rfloor \,\&\, ?\lfloor B\rfloor)$	$(\vee_I)$	$?\lfloor A \vee B\rfloor \otimes (?\lceil A\rceil \oplus ?\lceil B\rceil)$
$(\neg GI_1)$	$?\lfloor \neg A\rfloor \otimes ?\lfloor A\rfloor$	$(\neg GI_2)$	$?\lceil \neg A\rceil \otimes ?\lceil A\rceil$
(Id_1)	$\lfloor B\rfloor^{\perp} \otimes \lceil B\rceil^{\perp}$		

Figure 4.18: The specification $\mathcal{L}_{fd}$ for free deduction.

Proposition 4.16 *Let $\Gamma \cup \{C\}$ be a set of object-level formulas. Assume that all meta-level atomic formulas are given a negative polarity. Then $\Gamma \vdash^n C$ if and only if $\vdash_{llf} \mathcal{L}^n_{ge}, \lfloor\Gamma\rfloor : \lceil C\rceil \Uparrow$. Furthermore, adequacy for derivations also holds between the respective proof systems.*

Proof Proof by structural induction on the height of derivations. □

Proposition 4.17 *Let $\Gamma \cup \{C\}$ be a set of object logic formulas. Then*

$$\vdash_{ll} \mathcal{L}_J, Id_1, Str_L, W_R, ?\lfloor\Gamma\rfloor, \lceil C\rceil \text{ iff } \vdash_{ll} \mathcal{L}^n_{ge}, ?\lfloor\Gamma\rfloor, \lceil C\rceil$$

Proof This proposition is proved in a similar way as Proposition 4.4. □

The following corollary is a direct consequence of Propositions 4.2, 4.5, 4.16, and 4.17.

Corollary 4.18 *Let $\Gamma \cup \{C\}$ be a set of formulas. Then $\Gamma \vdash^n C$ if and only if $\Gamma \vdash^f_{lj} C$.*

4.6 Free Deduction

In [Parigot 1992], Parigot introduced the *free deduction* proof system for propositional classical logic that employed both the generalized elimination rules of the previous section and generalized introduction rules[1]. The inference rules for free deduction proof system are given in Figure 4.17. In order to treat classical negation here, we introduce the negation $\neg B$ directly here and do not treat it as an abbreviation for $B \Rightarrow \perp$.

We use the theory $\mathcal{L}_{fd}$ in Figure 4.18 to encode free deduction. To obtain the strongest level of adequacy, we assign negative polarity to all meta-level atoms. For example, the formula $(\neg GI_2)$

[1] Later and independently, Negri and von Plato also introduced generalized introduction rules in [Negri 2001, p. 214].

encodes the inference rule $\neg GI_2$, as is illustrated in the following derivation, where $\mathcal{K} = \mathcal{L}_{fd} \cup \lfloor\Gamma\rfloor \cup \lceil\Delta\rceil$:

$$\dfrac{\dfrac{\dfrac{\vdash \mathcal{K}, \lceil\neg A\rceil : \Uparrow}{\vdash \mathcal{K} : \Downarrow ?\lceil\neg A\rceil}\,[R\Uparrow, ?] \quad \dfrac{\vdash \mathcal{K}, \lceil A\rceil : \Uparrow}{\vdash \mathcal{K} : \Downarrow ?\lceil A\rceil}\,[R\Uparrow, ?]}{\vdash \mathcal{K} : \Downarrow ?\lceil\neg A\rceil \otimes ?\lceil A\rceil}\,[2\times\otimes]}{\vdash \mathcal{K} : \Uparrow}\,[D_2, \exists]$$

We can repeat this computation for all formulas in $\mathcal{L}_{fd}$ and in the process, prove the following proposition.

Proposition 4.19 *Let $\Gamma \cup \Delta$ be a set of object-level formulas. Assume that all meta-level atomic formulas are given a negative polarity. Then $\Gamma \vdash \Delta$ is provable in FD iff $\vdash \mathcal{L}_{fd}, \lfloor\Gamma\rfloor, \lceil\Delta\rceil : \Uparrow$ is provable in LLF. Furthermore, adequacy for derivations also holds between the respective proof systems.*

Proof The proof is by structural induction on the height of derivations. We show here some of the cases only, involving general introduction rules. The cases for the general elimination rules are similar to the corresponding cases in Proposition 4.11. In the derivation below $\mathcal{K} = \mathcal{L}_{fd} \cup \lfloor\Gamma\rfloor \cup \lceil\Delta\rceil$.

$$\dfrac{\Gamma, A \Rightarrow B \vdash \Delta \quad \Gamma, A \vdash \Delta}{\Gamma \vdash \Delta}\,[\Rightarrow GI_1] \quad \rightsquigarrow$$

$$\dfrac{\dfrac{\dfrac{\vdash \mathcal{K}, \lfloor A \Rightarrow B\rfloor : \cdot \Uparrow}{\vdash \mathcal{K} : \Downarrow ?\lfloor A \Rightarrow B\rfloor}\,[R\Uparrow, ?] \quad \dfrac{\dfrac{\vdash \mathcal{K}, \lfloor A\rfloor : \cdot \Uparrow}{\vdash \mathcal{K} : \Downarrow ?\lfloor A\rfloor}\,[R\Uparrow, ?]}{\vdash \mathcal{K} : \Downarrow ?\lfloor A\rfloor \oplus ?\lceil B\rceil}\,[\oplus_1]}{\vdash \mathcal{K} : \Downarrow ?\lfloor A \Rightarrow B\rfloor \otimes (?\lfloor A\rfloor \oplus ?\lceil B\rceil)}\,[\otimes]}{\vdash \mathcal{K} : \cdot \Uparrow}\,[D_2, 2\times\exists]$$

$$\dfrac{\Gamma, A \wedge B \vdash \Delta \quad \Gamma \vdash \Delta, A \quad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta}\,[\wedge GI] \quad \rightsquigarrow$$

$$\dfrac{\dfrac{\dfrac{\vdash \mathcal{K}, \lfloor A \wedge B\rfloor : \cdot \Uparrow}{\vdash \mathcal{K} : \Downarrow ?\lfloor A \wedge B\rfloor}\,[R\Uparrow, ?] \quad \dfrac{\dfrac{\vdash \mathcal{K}, \lceil A\rceil : \cdot \Uparrow}{\vdash \mathcal{K} : \Uparrow ?\lceil A\rceil}\,[?] \quad \dfrac{\vdash \mathcal{K}, \lceil B\rceil : \cdot \Uparrow}{\vdash \mathcal{K} : \Uparrow ?\lceil B\rceil}\,[?]}{\vdash \mathcal{K} : \Downarrow ?\lceil A\rceil \,\&\, ?\lceil B\rceil}\,[R\Uparrow, \&]}{\vdash \mathcal{K} : \Downarrow ?\lfloor A \wedge B\rfloor \otimes (?\lceil A\rceil \,\&\, ?\lceil B\rceil)}\,[\otimes]}{\vdash \mathcal{K} : \cdot \Uparrow}\,[D_2, 2\times\exists]$$

□

In order to relate the theory $\mathcal{L}_{fd}$ back to other theories, we must first replace $\neg$ by "implies false." We do this by using the operator ϕ inductively on propositional formulas as follows: $\phi(F \blacktriangle G) = \phi(F) \blacktriangle \phi(G)$; for all binary connectives $\blacktriangle$, $\phi(\neg F) = (\phi(F) \Rightarrow \bot)$; and $\phi(A) = A$ if A is an atom. Moreover, $\phi(\Gamma) = \{\phi(F) \mid F \in \Gamma\}$, where Γ is a multiset of formulas. We offer the following theorem as a means to related the provable formulas of $\mathcal{L}_{fd}$ with those in other classical theories.

Proposition 4.20 *Let $\Gamma \cup \Delta$ be a set of object logic, propositional classical formulas. Then*

$$\vdash_{ll} \mathcal{L}, Id_1, Id_2, Str_L, ?\lfloor\phi(\Gamma)\rfloor, ?\lceil\phi(\Delta)\rceil \quad \textit{iff} \quad \vdash_{ll} \mathcal{L}_{fd}, ?\lfloor\Gamma\rfloor, ?\lceil\Delta\rceil.$$

Proof The $\Leftarrow$ direction is proved in similar way as Proposition 4.3, by using the equivalences obtained from the structural and identity rules.

The $\Rightarrow$ direction is proved in similar way as in Proposition 4.3, by assigning negative polarity to the meta-level atoms. However, for the inductive case when the clause Id_2 is focused on, we use Parigot's observation that any instance of a sequent calculus cut-rule is translated in Free Deduction to a sequence of elimination and introduction rules whose main premises is the cut-formula. □

From Propositions 4.3 and 4.20, we have the following relationship between sequents provable in free deduction and those provable in the LK sequent calculus.

Corollary 4.21 *Let Γ and Δ be sets of propositional, classical formulas. Then $\Gamma \vdash \Delta$ is provable in FD if and only if $\phi(\Gamma) \vdash \phi(\Delta)$ is provable in LK^c.*

Parigot notes that if one of the premises of the generalized rules is "killed", *i.e.*, it is always the conclusion of an initial rule, then one can obtain either sequent calculus or natural deduction proofs with multiple conclusions. The "killing" of a premise is accounted for in our framework by the use of polarities to enforce the presence of a formula in the context of the sequent. Our encoding of the LK calculus could be explained by just using such a focusing restriction. A presentation of a natural deduction with multiple conclusions could be obtained in a similar way as for the natural deduction with single conclusion but with the main difference being that one has to also incorporate the Str_R rule in the theory by adding ? to positive occurrences of $\lceil\cdot\rceil$ atoms and negative occurrences of $\lfloor\cdot\rfloor$ atoms.

4.7 The Tableaux Proof System KE

In the previous sections, we dealt with systems that contained rules with more premises than the corresponding rules in sequent calculus or natural deduction. Now, we move to the other direction and deal with systems that contain rules with fewer premises.

In [D'Agostino 1994], D'Agostino and Mondadori proposed the propositional tableaux system KE displayed in Figure 4.19. Here, the only rule that has more than one premise is the cut rule. In the original system, the cut inference rule appears with a side condition limiting cuts to be analytical cuts, that is, rules where the cut-formula is a subformula of a formula in the end-sequent: since that condition does not seem to be treated naturally in our context, we consider only the unrestricted cut rule.

To encode KE, we use the theory $\mathcal{L}_{ke}$ in Figure 4.20. To obtain an adequacy on the level of derivations from $\mathcal{L}_{ke}$, we assign negative polarity to all atoms $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$. As before, the negative occurrences of $\lceil\cdot\rceil$ and $\lfloor\cdot\rfloor$ enforce the presence of formulas in the sequent, but now, $\mathcal{L}_{ke}$ contains formulas with two negative occurrences of meta-level atoms. These formulas encode the KE rules that contain only one premise. For example, the clause $(\Rightarrow_{L2})$ encodes KE's inference rule $\Rightarrow_{L2}$, as illustrates the following derivation, where $\mathcal{K} = \mathcal{L}_{ke} \cup \lfloor \Gamma, A \Rightarrow B \rfloor \cup \lceil \Delta, B \rceil$:

$$\dfrac{\dfrac{\dfrac{}{\vdash \mathcal{K} :\Downarrow \lfloor A \Rightarrow B \rfloor^{\perp}}\,[I_2] \quad \dfrac{\vdash \mathcal{K}, \lceil A \rceil : \cdot \Uparrow}{\vdash \mathcal{K} :\Downarrow ?\lceil A \rceil}\,[R\Downarrow, ?] \quad \dfrac{}{\vdash \mathcal{K} :\Downarrow \lceil B \rceil^{\perp}}\,[I_2]}{\vdash \mathcal{K} :\Downarrow \lfloor A \Rightarrow B \rfloor^{\perp} \otimes (?\lceil A \rceil \otimes \lceil B \rceil^{\perp})}\,[2\times\otimes]}{\vdash \mathcal{K} : \cdot \Uparrow}\,[D_2, 2\times\exists]$$

By checking all the other the inference rules generated by focusing on formulas in $\mathcal{L}_{ke}$, we can conclude with the following proposition.

Proposition 4.22 *Let $\Gamma \cup \Delta$ be a set of object-level formulas. Assume that all meta-level atomic formulas are given a negative polarity. Then $\Gamma \vdash \Delta$ is provable in KE iff $\vdash \mathcal{L}_{ke}, \lfloor\Gamma\rfloor, \lceil\Delta\rceil :\Uparrow$ is provable in LLF.*

Proof The proof is by structural induction on the height of derivations. We show only some of the cases. In the derivations below $\mathcal{K} = \mathcal{L}_{ke} \cup \lfloor\Gamma\rfloor \cup \lceil\Delta\rceil$.

$$\frac{\Gamma, A, A \Rightarrow B, B \vdash \Delta}{\Gamma, A, A \Rightarrow B \vdash \Delta}\ [\Rightarrow_{L1}] \quad \frac{\Gamma, A \Rightarrow B \vdash A, B, \Delta}{\Gamma, A \Rightarrow B \vdash B, \Delta}\ [\Rightarrow_{L2}] \quad \frac{\Gamma, A \vdash A \Rightarrow B, B, \Delta}{\Gamma \vdash A \Rightarrow B, \Delta}\ [\Rightarrow_{R}]$$

$$\frac{\Gamma, A \wedge B, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\ [\wedge_{L}] \quad \frac{\Gamma, A \vdash A \wedge B, B, \Delta}{\Gamma, A \vdash A \wedge B, \Delta}\ [\wedge_{R1}] \quad \frac{\Gamma, B \vdash A \wedge B, A, \Delta}{\Gamma, B \vdash A \wedge B, \Delta}\ [\wedge_{R2}]$$

$$\frac{\Gamma, A \vee B, B \vdash A, \Delta}{\Gamma, A \vee B \vdash A, \Delta}\ [\vee_{L1}] \quad \frac{\Gamma, A \vee B, A \vdash B, \Delta}{\Gamma, A \vee B \vdash B, \Delta}\ [\vee_{L2}] \quad \frac{\Gamma \vdash A, B, A \vee B, \Delta}{\Gamma \vdash A \vee B, \Delta}\ [\vee_{R}]$$

$$\frac{\Gamma, \neg A \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta}\ [\neg_{L}] \quad \frac{\Gamma, A \vdash \neg A, \Delta}{\Gamma \vdash \neg A, \Delta}\ [\neg_{R}]$$

$$\frac{}{\Gamma, A \vdash A, \Delta}\ [\mathrm{I}] \quad \frac{\Gamma, A \vdash \Delta \quad \Gamma \vdash A, \Delta}{\Gamma \vdash \Delta}\ [\mathrm{Cut}]$$

Figure 4.19: The rules for the classical propositional logic KE.

$(\Rightarrow_{L1})$	$\lfloor A \Rightarrow B \rfloor^{\perp} \otimes (\lfloor A \rfloor^{\perp} \otimes ?\lfloor B \rfloor)$	$(\Rightarrow_{R})$	$\lceil A \Rightarrow B \rceil^{\perp} \otimes (?\lfloor A \rfloor \parr ?\lceil B \rceil)$
$(\Rightarrow_{L2})$	$\lfloor A \Rightarrow B \rfloor^{\perp} \otimes (?\lceil A \rceil \otimes \lceil B \rceil^{\perp})$	$(\wedge_{R1})$	$\lceil A \wedge B \rceil^{\perp} \otimes (\lfloor A \rfloor^{\perp} \otimes ?\lceil B \rceil)$
$(\wedge_{L})$	$\lfloor A \wedge B \rfloor^{\perp} \otimes (?\lfloor A \rfloor \parr ?\lfloor B \rfloor)$	$(\wedge_{R2})$	$\lceil A \wedge B \rceil^{\perp} \otimes (?\lceil A \rceil \otimes \lfloor B \rfloor^{\perp})$
$(\vee_{L1})$	$\lfloor A \vee B \rfloor^{\perp} \otimes (\lceil A \rceil^{\perp} \otimes ?\lfloor B \rfloor)$	$(\vee_{R})$	$\lceil A \vee B \rceil^{\perp} \otimes (?\lceil A \rceil \parr ?\lceil B \rceil)$
$(\vee_{L2})$	$\lfloor A \vee B \rfloor^{\perp} \otimes (\lfloor A \rfloor \otimes \lceil B \rceil^{\perp})$		
$(\neg_{L})$	$\lfloor \neg A \rfloor^{\perp} \otimes \lceil A \rceil$	$(\neg_{R})$	$\lceil \neg A \rceil^{\perp} \otimes \lfloor A \rfloor$
(Id_1)	$\lfloor B \rfloor^{\perp} \otimes \lceil B \rceil^{\perp}$	(Id_2)	$?\lfloor B \rfloor \otimes ?\lceil B \rceil$

Figure 4.20: The specification $\mathcal{L}_{ke}$ for the system KE.

$$\frac{\Gamma, A \wedge B, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\,[\wedge_L] \quad \rightsquigarrow$$

$$\dfrac{\dfrac{}{\vdash \mathcal{K}, \lfloor A \wedge B \rfloor :\Downarrow \lfloor A \wedge B \rfloor^{\perp}}\,[I_2] \quad \dfrac{\dfrac{\vdash \mathcal{K}, \lfloor A \wedge B \rfloor, \lfloor A \rfloor, \lfloor B \rfloor :\Uparrow}{\vdash \mathcal{K}, \lfloor A \wedge B \rfloor :\Uparrow ?\lfloor A \rfloor, ?\lfloor B \rfloor}\,[2 \times ?]}{\vdash \mathcal{K}, \lfloor A \wedge B \rfloor :\Downarrow ?\lfloor A \rfloor \parr ?\lfloor B \rfloor}\,[R\Uparrow, \parr]}{\dfrac{\vdash \mathcal{K}, \lfloor A \wedge B \rfloor :\Downarrow \lfloor A \wedge B \rfloor^{\perp} \otimes (?\lfloor A \rfloor \parr ?\lfloor B \rfloor)}{\vdash \mathcal{K}, \lfloor A \wedge B \rfloor : \cdot \Uparrow}\,[D_2, 2 \times \exists]}\,[\otimes]$$

$$\frac{\Gamma, A \vee B, B \vdash A, \Delta}{\Gamma, A \vee B \vdash A, \Delta}\,[\vee_{L1}] \quad \rightsquigarrow$$

$$\dfrac{\dfrac{}{\vdash \mathcal{K}, \lfloor A \vee B \rfloor, \lceil A \rceil :\Downarrow \lfloor A \vee B \rfloor^{\perp}}\,[I_2] \quad \dfrac{}{\vdash \mathcal{K}, \lfloor A \vee B \rfloor, \lceil A \rceil :\Downarrow \lceil A \rceil^{\perp}}\,[I_2] \quad \dfrac{\mathcal{K}, \lfloor A \vee B \rfloor, \lfloor B \rfloor, \lceil A \rceil : \cdot \Uparrow}{\vdash \mathcal{K}, \lfloor A \vee B \rfloor, \lceil A \rceil :\Downarrow ?\lfloor B \rfloor}\,[R\Uparrow, ?]}{\dfrac{\vdash \mathcal{K}, \lfloor A \vee B \rfloor, \lceil A \rceil :\Downarrow \lfloor A \vee B \rfloor^{\perp} \otimes (\lceil A \rceil^{\perp} \otimes ?\lfloor B \rfloor)}{\vdash \mathcal{K}, \lfloor A \vee B \rfloor, \lceil A \rceil : \cdot \Uparrow}\,[D_2, 2 \times \exists]}\,[2 \times \otimes]$$

□

The following proposition is proved by induction on the height of proofs, by taking into consideration the equivalences obtained by the identity and structural rules, and by using the operator ϕ to replace $\neg$ in formulas by its "implies false" meaning.

Proposition 4.23 *Let $\Gamma \cup \Delta$ be a set of object logic, classical, propositional formulas. Then*

$$\vdash_{ll} \mathcal{L}, Id_1, Id_2, Str_L, Str_R, ?\lfloor \phi(\Gamma) \rfloor, ?\lceil \phi(\Delta) \rceil \quad \textit{iff} \quad \vdash_{ll} \mathcal{L}_{ke}, ?\lfloor \Gamma \rfloor, ?\lceil \Delta \rceil$$

Proof The proof is similar to the proof of Lemma 4.3. □

The following result, establishing the equivalence between KE and propositional LK^c, is a direct consequence of Propositions 4.1, 4.3, 4.22 and 4.23.

Corollary 4.24 *Let Γ and Δ be a set of propositional formulas. Then $\Gamma \vdash \Delta$ is provable in KE if and only if $\phi(\Gamma) \vdash_{lk} \phi(\Delta)$ is provable in the propositional fragment of LK^c.*

4.8 Smullyan's Analytic Cut System

To illustrate how one can capture another extreme in proof systems, we consider Smullyan's proof system for analytic cut (AC) [Smullyan 1968a], which is depicted in Figure 4.21. Here, all rules except the cut rule have no premises. As the name of the system suggests, Smullyan also assigned a side condition to the cut rule, allowing only analytical cuts. As in the previous section, we shall drop this restriction as it is not directly captured in our framework.

We again assign negative polarity to $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ atoms and use the theory $\mathcal{L}_{ac}$, shown in Figure 4.22, to obtain the strongest level of adequacy. For example, the formula $(\Rightarrow_L)$ corresponds to the inference rule $\Rightarrow_L$ in AC, as illustrates the following derivation, where $\mathcal{K} = \mathcal{L}_{ac} \cup \lfloor \Gamma \rfloor \cup \lceil \Delta \rceil$ such that $A \Rightarrow B, A \in \Gamma$ and $B \in \Delta$:

$$\dfrac{\dfrac{}{\vdash \mathcal{K} : \cdot \Downarrow \lfloor A \Rightarrow B \rfloor^{\perp}}\,[I_2] \quad \dfrac{}{\vdash \mathcal{K} : \cdot \Downarrow \lfloor A \rfloor^{\perp}}\,[I_2] \quad \dfrac{}{\vdash \mathcal{K} : \cdot \Downarrow \lceil B \rceil^{\perp}}\,[I_2]}{\dfrac{\vdash \mathcal{K} : \cdot \Downarrow \lfloor A \Rightarrow B \rfloor^{\perp} \otimes (\lfloor A \rfloor^{\perp} \otimes \lceil B \rceil^{\perp})}{\vdash \mathcal{K} : \cdot \Uparrow}\,[D_2, 2 \times \exists]}\,[2 \times \otimes]$$

Again, the following proposition follows from repeating such constructions for all formulas in $\mathcal{L}_{ac}$.

$$\frac{}{\Gamma, A \vee B \vdash A, B, \Delta}\ [\vee_L] \qquad \frac{}{\Gamma, A \vdash A \vee B, \Delta}\ [\vee_{R1}] \qquad \frac{}{\Gamma, B \vdash A \vee B, \Delta}\ [\vee_{R2}]$$

$$\frac{}{\Gamma, A \wedge B \vdash A, \Delta}\ [\wedge_{L1}] \qquad \frac{}{\Gamma, A \wedge B \vdash B, \Delta}\ [\wedge_{L2}] \qquad \frac{}{\Gamma, A, B \vdash A \wedge B, \Delta}\ [\wedge_R]$$

$$\frac{}{\Gamma, A, A \Rightarrow B \vdash B, \Delta}\ [\Rightarrow_L] \qquad \frac{}{\Gamma \vdash A, A \Rightarrow B, \Delta}\ [\Rightarrow_{R1}] \qquad \frac{}{\Gamma, B \vdash A \Rightarrow B, \Delta}\ [\Rightarrow_{R2}]$$

$$\frac{}{\Gamma, \neg A, A \vdash \Delta}\ [\neg_L] \qquad \frac{}{\Gamma \vdash A, \neg A, \Delta}\ [\neg_R]$$

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma \vdash A, \Delta}{\Gamma \vdash \Delta}\ [\text{Cut}] \qquad \frac{}{\Gamma, A \vdash A, \Delta}\ [\text{I}]$$

Figure 4.21: Smullyan's Analytic Cut System for classical propositional logic, AC, except that the cut rule is not restricted.

$(\Rightarrow_L)$	$\lfloor A \Rightarrow B \rfloor^\perp \otimes (\lfloor A \rfloor^\perp \otimes \lceil B \rceil^\perp)$	$(\Rightarrow_R)$	$\lceil A \Rightarrow B \rceil^\perp \otimes (\lceil A \rceil^\perp \oplus \lfloor B \rfloor^\perp)$
$(\wedge_L)$	$\lfloor A \wedge B \rfloor^\perp \otimes (\lceil A \rceil^\perp \oplus \lceil B \rceil^\perp)$	$(\wedge_R)$	$\lceil A \wedge B \rceil^\perp \otimes (\lfloor A \rfloor^\perp \otimes \lfloor B \rfloor^\perp)$
$(\vee_L)$	$\lfloor A \vee B \rfloor^\perp \otimes (\lceil A \rceil^\perp \otimes \lceil B \rceil^\perp)$	$(\vee_R)$	$\lceil A \vee B \rceil^\perp \otimes (\lfloor A \rfloor^\perp \oplus \lfloor B \rfloor^\perp)$
$(\neg_L)$	$\lceil \neg A \rceil^\perp \otimes \lceil A \rceil^\perp$	$(\neg_R)$	$\lfloor \neg A \rfloor^\perp \otimes \lfloor A \rfloor^\perp$
(Id_1)	$\lfloor B \rfloor^\perp \otimes \lceil B \rceil^\perp$	(Id_2)	$?\lfloor B \rfloor \otimes ?\lceil B \rceil$

Figure 4.22: The theory $\mathcal{L}_{ac}$ used to encode Smullyan's Analytic Cut System AC.

Proposition 4.25 *Let $\Gamma \cup \Delta$ be a set of object-level, classical propositional formulas. Assume that all meta-level atomic formulas are given a negative polarity. Then $\Gamma \vdash \Delta$ is provable in AC iff $\vdash \mathcal{L}_{ac}, \lfloor \Gamma \rfloor, \lceil \Delta \rceil : \Uparrow$ is provable in LLF. Furthermore, adequacy for derivations also holds between the respective proof systems.*

Proof The proof is by structural induction on the height of derivations. We show only some of the remaining cases.

$$\frac{}{\Gamma, A \vee B \vdash A, B, \Delta}\ [\vee_L] \quad \rightsquigarrow \quad \frac{\dfrac{\dfrac{}{\vdash \mathcal{K}_\vee : \cdot \Downarrow \lfloor A \vee B \rfloor^\perp}\ [I_2] \quad \dfrac{}{\vdash \mathcal{K}_\vee : \cdot \Downarrow \lceil A \rceil^\perp}\ [I_2] \quad \dfrac{}{\vdash \mathcal{K}_\vee : \cdot \Downarrow \lceil B \rceil^\perp}\ [I_2]}{\vdash \mathcal{K}_\vee : \cdot \Downarrow \lfloor A \vee B \rfloor^\perp \otimes (\lceil A \rceil^\perp \otimes \lceil B \rceil^\perp)}\ [2 \times \otimes]}{\vdash \mathcal{K}_\vee : \cdot \Uparrow}\ [D_2, 2 \times \exists]$$

where $\mathcal{K}_\vee = \mathcal{L}_{ac} \cup \lfloor \Gamma, A \vee B \rfloor \cup \lceil \Delta, A, B \rceil$.

$$\frac{}{\Gamma, A \wedge B \vdash A, \Delta}\ [\wedge_{L1}] \quad \rightsquigarrow \quad \frac{\dfrac{\dfrac{}{\vdash \mathcal{K}_\wedge : \cdot \Downarrow \lfloor A \wedge B \rfloor^\perp}\ [I_2] \quad \dfrac{\dfrac{}{\vdash \mathcal{K}_\wedge : \cdot \Downarrow \lceil A \rceil^\perp}\ [I_2]}{\vdash \mathcal{K}_\wedge : \cdot \Downarrow \lceil A \rceil^\perp \oplus \lceil B \rceil^\perp}\ [\oplus_1]}{\vdash \mathcal{K}_\wedge : \cdot \Downarrow \lfloor A \wedge B \rfloor^\perp \otimes (\lceil A \rceil^\perp \oplus \lceil B \rceil^\perp)}\ [\otimes]}{\vdash \mathcal{K}_\wedge : \cdot \Uparrow}\ [D_2, 2 \times \exists]$$

where $\mathcal{K}_\wedge = \mathcal{L}_{ac} \cup \lfloor \Gamma, A \wedge B \rfloor \cup \lceil \Delta, A \rceil$. $\square$

Again by using the equalities obtained from the identity and structural rules and the operator ϕ, we obtain the following proposition.

Proposition 4.26 *Let $\Gamma \cup \Delta$ be a set of object logic, classical propositional formulas. Then*

$$\vdash_{ll} \mathcal{L}, Id_1, Id_2, Str_L, Str_R, ?\lfloor \phi(\Gamma) \rfloor, ?\lceil \phi(\Delta) \rceil \quad \textit{iff} \quad \vdash_{ll} \mathcal{L}_{ac}, ?\lfloor \Gamma \rfloor, ?\lceil \Delta \rceil$$

Proof The proof is similar to the proof of Proposition 4.3. □

The following result follows directly from the Propositions 4.1, 4.3, 4.25, and 4.26.

Corollary 4.27 *Let Γ and Δ be a set of classical, propositional formulas. Then $\Gamma \vdash \Delta$ is provable in AC if and only if $\phi(\Gamma) \vdash \phi(\Delta)$ is provable in the propositional fragment of LK^c.*

4.9 Related Work

A number of logical frameworks have been proposed to represent object-level proof systems. Many of these frameworks, as used in [Felty 1988, Harper 1993, Pfenning 1989], are based on intuitionistic (minimal) logic principles. In such settings, the dualities that we employ here, for example, $\lfloor B \rfloor \equiv \lceil B \rceil^{\perp}$, are not available within the logic and this makes reasoning about the relative completeness between object-level proof systems harder. Also, since minimal logic sequents must have a single conclusion, the storage of object-level formulas is generally done on the left-hand side of meta-level sequents (see [Hodas 1994a, Pfenning 2000]) with some kind of "marker" for the right-hand side (such as the non-logical "refutation" marker # in [Pfenning 2000]). The flexibility of having the four meta-level literals $\lfloor B \rfloor$, $\lceil B \rceil$, $\lfloor B \rfloor^{\perp}$, and $\lceil B \rceil^{\perp}$ is not generally available in such intuitionistic systems. While it is natural in classical linear logic to consider having some atoms assigned negative and some positive polarities, most intuitionistic systems consider only uniform assignments of polarities to meta-level atoms (usually negative in order to support goal-directed proof search): the ability to mix polarity assignments for different meta-level atoms can only be achieved in more indirect fashions in such settings.

The abstract logic programming presentation of linear logic called Forum [Miller 1996] has been used to specify sequent calculus proof systems in a style similar to that used here. That presentation of linear logic was, however, also limited in that negation was not a primitive connective and that all atomic formulas were assumed to have negative polarity. The range of encodings contained in this chapter are not directly available using Forum.

In [Ciabattoni 2008], Ciabattoni *et al.* consider a general approach to the specification of structural rules in sequent calculus which differs from our approach of specifying structural rules. In particular, their method would not use the exponentials of linear logic, as we do in the clauses Str_L and Str_R, but would rather treat structural rules more explicitly by having rules of the form

$$\lfloor B \rfloor^{\perp} \otimes (\lfloor B \rfloor \parr \lfloor B \rfloor)$$

to encode the contraction-left rules (of the sequent calculus).

4.10 Conclusions and further remarks

We have shown that by employing different focusing annotations or using different sets of formulas that are (meta-logically) equivalent to $\mathcal{L}$, a range of sound and (relatively) complete object-level proof systems can be encoded. We have illustrated this principle by showing how linear logic focusing and logical equivalences can account for object-level proof systems based on sequent calculus, natural deduction, generalized introduction and elimination rules, free deduction, the tableaux system KE, and Smullyan's AC system employing only axioms and the cut rule.

We now point out some directions for future work:

- *Size of proofs* – An interesting line of future research would be to consider differences in the *sizes* of proofs in these different paradigms since these differences can be related to the topic of comparing bottom-up and top-down deduction. Thus, it might be possible to flexibly change polarity assignments that would result in different and, hopefully, more compact presentations of proofs.

- *Transform proofs between different systems* – Although we used the same theory to encode (open) derivations of different proof systems, we did not investigate how to transform a proof in one system to a proof in another system. These transformations would have to deal not only with the use of equivalent formulas, such as using additive or multiplicative conjunctions, but also deal with the change in the polarity assignment of the meta-level literals.

  One way of transforming a focused proof with one polarity assignment to another focused proof with a different polarity assignment is by using foculisation graphs (see Section 2.5). Let Ξ be a focused proof with a polarity assignment. We first remove the focusing annotations from Ξ, obtaining hence, the proof Ξ'. Then we transform Ξ' into a focused proof with a different polarity assignment by using the foculisation graphs obtained with this new polarity assignment.

- *Encoding natural deduction normal form proofs* – In this chapter, we have only been able to encode the natural deduction normal form proofs that did not contain disjunctions and existential quantifiers. When these connectives are allowed, normal form proofs contain not only an introduction phase and an elimination phase, but also a phase between these two phases called *segment*. The problem of encoding these type of proofs is that the encoding of the existential and disjunction elimination rules does not enforce that these rules appear between the introduction and elimination phase. However, it is easy to check that the derivations encoding the other elimination rules permute over the derivations encoding the elimination rules for existential quantifiers and disjunctions. If one could permute downwards all derivations encoding disjunction and existential elimination rules, we would be able to encode normal form proofs. A way to do so is to use the notion of maximally multifocused proofs [Chaudhuri 2008a]. Multifocusing is a trivial generalization of focused systems, where not only one formula can be focused on, but a set of formulas can be focused. Chaudhuri *et al.* showed the existence of maximally multifocused proofs where one focuses on the multiset containing the most formulas. We speculate that normal form proofs would correspond to maximally multifocused proofs.

- *Analytic cuts* – We were not yet able to give a logical account for the side condition on analytic cut-rules. The answer for this problem does not seem easy. One way to do so, is to express the subformula condition implicitly by using meta-level initial rules. For example, one could use special tokens for analytic cut formulas, such as $\lfloor A \rfloor_a$ and $\lceil A \rceil_a$, that would have to necessarily "match" subformulas, $\lfloor A \rfloor$ and $\lceil A \rceil$, of any formula in the endsequent in an meta-level initial rule. Although this type of approach might work to express complete set of proofs for KE and AC, there are at least two problems with this approach. First, we cannot encode all proofs of AC and of KE, "only" a complete set of proofs. Second, one cannot check locally that the cut-formula used satisfies the subformula condition, but only when the encoding of the analytic cut is used in a meta-level initial.

- *More flexible polarity assignments* – We have only considered a global polarity assignment for $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ meta-level atoms. One could imagine more flexible polarity assignments, where some occurrences of $\lfloor \cdot \rfloor$ (resp. $\lceil \cdot \rceil$) are assigned positive polarity and other occurrences negative polarity. Since changing polarity assignments does not affect provability one would obtain yet another complete proof system. Another future work is to find good applications for such proof systems.

While focusing at the meta-level clearly provides a powerful normal form of proof, we have not described how to use the techniques presented in this chapter to derive object-level focusing proof systems. Finding a means to derive such object-level normal form proofs is an interesting challenge and we will tackle this problem in Chapter 6.

Chapter 5

Linear Logic with subexponentials

It is well known that the linear logic exponentials are not canonical: suppose that we have two pairs of exponentials; one pair colored as blue, $!^b$ and $?^b$, and the second pair colored as red, $!^r$ and $?^r$, with the following introduction rules:

$$\frac{\vdash ?^b\Gamma, F}{\vdash ?^b\Gamma, !^b F}\,[!^b] \qquad \frac{\vdash \Gamma, F}{\vdash \Gamma, ?^b F}\,[D?^b] \qquad \frac{\vdash ?^r\Gamma, F}{\vdash ?^r\Gamma, !^r F}\,[!^r] \qquad \frac{\vdash \Gamma, F}{\vdash \Gamma, ?^r F}\,[D?^r]$$

One cannot show that $!^r F \equiv !^b F$ for any formula F.

In fact, it is possible to fill the gap between the "linear" modality and the ? modality in linear logic with a preordered set of exponential-like operators. These intermediate modalities may or may not permit weakening and contraction. We introduce the term *subexponential* for such modalities since the exponential equation $?(A \oplus B) \equiv (?A \parr ?B)$ fails when contraction and weakening are not admitted for ?. In this chapter, we discuss linear logic proof systems that contain subexponentials.

The remainder of this chapter is structured as follows: in Section 5.1 we introduce the linear logic system *SELL* with *subexponentials*. Then in Section 5.2 we propose some focusing systems for *SELL*, first considering the case where *relevant* formulas, which can contract but not weaken, are not permitted, and then we consider the general case where such formulas are also permitted. In Section 5.3, we propose an extension to *SELL* where subexponentials can be instantiated or created, and propose a focused version of such a system. Finally, in Sections 5.4 and 5.5 we finish the chapter by commenting on related works and pointing out directions for future work.

References: parts of this chapter are in [Nigam 2009a].

5.1 Linear Logic with Subexponentials

Danos *et al.* proposed in [Danos 1993] a linear logic system that contains non-canonical connectives for exponentials which we refer to as *subexponentials*. A *subexponential signature* is the tuple $\langle \mathcal{I}, \preceq, \mathcal{W}, \mathcal{C}\rangle$ where $\mathcal{I}$ is a set of indexes, with the "colors" of the subexponentials, $\preceq$ is a preordered relation[1] over the elements of the set of indexes $\mathcal{I}$, and $\mathcal{W}$ and $\mathcal{C}$ are both subsets of $\mathcal{I}$. Given a subexponential signature $\Sigma = \langle \mathcal{I}, \preceq, \mathcal{W}, \mathcal{C}\rangle$, the rules for $SELL_\Sigma$ are the same as in linear logic, except that we add the following rules for the subexponentials.

- For each $a \in \mathcal{I}$, add the usual dereliction rule for $?^a$;

$$\frac{\vdash C, \Delta}{\vdash ?^a C, \Delta}\,[D?^a]$$

- For each $a \in \mathcal{W}$, add the usual weakening rule for $?^a$; and for each $b \in \mathcal{C}$, add the usual contraction rule for $?^b$;

$$\frac{\vdash \Delta}{\vdash ?^a C, \Delta}\,[W?^a] \qquad \frac{\vdash ?^b C, ?^b C, \Delta}{\vdash ?^b C, \Delta}\,[C?^b]$$

[1] A preorder relation is a binary relation that is reflexive ($a \preceq a$) and transitive (if $a \preceq b$ and $b \preceq c$ then $a \preceq c$).

- Add the following promotion rule for all $a \in \mathcal{I}$:

$$\frac{\vdash ?^{x_1}C_1, \ldots, ?^{x_n}C_n, C}{\vdash ?^{x_1}C_1, \ldots, ?^{x_n}C_n, !^a C}\ [!^a]$$

  with the proviso that $a \preceq x_i$ for all $i = 1, \ldots, n$.

The sets $\mathcal{W}$ and $\mathcal{C}$ in the subexponential signature specify which subexponential indexes in $\mathcal{I}$ allow for weakening and contraction, respectively. The preorder $\preceq$ is used only in the promotion rule. One can only introduce a subexponential bang of index a (for short, a-bang), if there are only formulas whose main connective is a subexponential question-mark of index x (for short, x-question mark) where $a \preceq x$. Hence, from an operational point of view, one must check if all indexes x are greater than a. We will elide the subscript Σ from $SELL_\Sigma$ whenever the subexponential signature Σ is clear from the context. Moreover, notice that when the subexponential signature is of the form $\langle \{u\}, \preceq, \{u\}, \{u\} \rangle$, where the preorder is the trivial identity relation, then $SELL_\Sigma$ corresponds exactly to the usual linear logic system. As we will point out later, this restriction over the subexponential signatures is necessary to prove the cut-elimination theorem for such logics.

For any subexponential signature Σ, the De Morgan's laws for the subexponentials in Σ are provable in $SELL_\Sigma$

$$(!^i F)^\perp \equiv ?^i F^\perp \qquad\qquad (?^i F)^\perp \equiv !^i F^\perp$$

As usual, we denote by $F^\perp$ the negation normal form of F, obtained by using the De Morgan's laws to push the negation inside the formula.

Since some subexponentials may or may not allow contractions and weakening of formulas, the equivalences, which give *exponentials* their names, are not necessarily provable in *SELL*, but only some directions, hence the name *subexponentials*. Consider the following subexponential indexes $c \in \mathcal{C}$, $w \in \mathcal{W}$ and $u \in \mathcal{W} \cap \mathcal{C}$, then the following formulas are provable in *SELL*, for any formulas P and Q:

- $!^c(P \,\&\, Q) \multimap (!^c P) \otimes (!^c Q)$;
- $?^w(P \oplus Q) \multimap (?^w P) \parr (?^w Q)$;
- $!^u(P \,\&\, Q) \equiv (!^u P) \otimes (!^u Q)$;
- $(!^w P) \otimes (!^w Q) \multimap !^w(P \,\&\, Q)$;
- $(?^c P) \parr (?^c Q) \multimap ?^c(P \oplus Q)$;
- $?^u(P \oplus Q) \equiv (?^u P) \parr (?^u Q)$.

We also classify formulas into four different groups: we classify as *linear* or *bounded* the formulas that are not allowed to contract nor weaken; as *affine* the formulas that are not allowed to contract but are allowed to weaken; as *relevant* the formulas that are allowed to contract, but not allowed to weaken; and as *unbounded* the formulas that are allowed to weaken and to contract.

Danos, Joinet and Schellinx showed in [Danos 1993] that the cut rule is admissible in $SELL_\Sigma$ if the subexponential signature satisfies the following condition: (i) if $i \in \mathcal{C}$ (resp. $i \in \mathcal{W}$) then for all j such that $i \preceq j$, j must also belong to $\mathcal{C}$ (resp. $\mathcal{W}$). That is, the subexponential indexes are closed upwardly over contraction and weakening. From now on we only consider such subexponential signatures. Moreover, they also showed that because of the reflexivity of the preorder relation, one can use only atomic initial rules.

Definition 5.1 A subexponential signature, $\Sigma = \langle \mathcal{I}, \preceq, \mathcal{W}, \mathcal{C} \rangle$, is consistent if for all $i, j \in \mathcal{I}$, if $i \in \mathcal{W}$ (respectively $\mathcal{C}$) and $i \preceq j$, then $j \in \mathcal{W}$ (respectively $\mathcal{C}$).

Proposition 5.2 *Let Σ be a consistent subexponential signature. A formula F is provable in $SELL_\Sigma$ if and only if it is provable in $SELL_\Sigma$ by only using instances of atomic initial rules.*

Proof As in the proof of Proposition 2.1, we use transformations of the form:

$$\frac{}{\vdash !^a F, ?^a F^\perp}\ [\mathrm{I}] \qquad \rightsquigarrow \qquad \frac{\dfrac{}{\vdash F, F^\perp}\ [D?^a]}{\vdash !^a F, ?^a F^\perp}\ [!^a]$$

Notice that this transformation works only because the relation $\preceq$ is reflexive. □

Theorem 5.3 *Let Σ be a consistent subexponential signature. Then, the cut elimination theorem holds for $SELL_\Sigma$.*

Proof The proof is similar to the proof of cut elimination in classical linear logic. We show here only the cases involving the subexponential question-marks and bangs. First we show that, whenever a rule is applied to a formula in the right (or left) premise of a cut rule, the cut rule can permute over this rule. For example:

$$\dfrac{\dfrac{\vdash \Gamma, P}{\vdash \Gamma, !^a P} \quad \dfrac{\vdash \Delta, Q, ?^a P^\perp}{\vdash \Delta, !^b Q, ?^a P^\perp}}{\vdash \Gamma, \Delta, !^b Q} \quad \rightsquigarrow \quad \dfrac{\dfrac{\vdash \Gamma, P}{\vdash \Gamma, !^a P} \quad \vdash \Delta, Q, ?^a P^\perp}{\dfrac{\vdash \Gamma, \Delta, Q}{\vdash \Gamma, \Delta, !^b Q}}$$

This transformation is valid because the relation $\preceq$ in Σ is a preorder. From the left branch of the cut, we know that Γ contains formulas whose main connective is a x-question mark such that $a \preceq x$, and from the right-premise, it must be the case that $b \preceq a$ and that Δ contains formulas whose main connective is a y-question mark such that $b \preceq y$. From transitivity of the preorder $\preceq$ it is the case that $b \preceq x$, and therefore the permutation works.

Now, we proceed with the elimination of cut. As usual, we first specify the degree $\delta(A)$ of a formula A, inductively, as follows:

1. $\delta(A) = 1$, if A is a literal;
2. $\delta(A \otimes B) = \delta(A \parr B) = \delta(A \,\&\, B) = \delta(A \oplus B) = max\{\delta(A), \delta(B)\} + 1$;
3. $\delta(!^i A) = \delta(?^i A) = \delta(A) + 1$, where i is a subexponential index.

The degree of a cut is defined to be the degree of the cut-formula, and the degree of a proof the supremum of the degrees of the cuts in the proof.

We show that one can reduce the degree of a proof by performing the following key transformations. Here, we also assume that the degree of the subtrees have degree strictly less than the degree of the proof-tree.

One can replace, for any subexponential index i, the derivation to the left by the derivation to the right with lower degree:

$$\dfrac{\dfrac{\vdash \Gamma, A}{\vdash \Gamma, !^i A} \quad \dfrac{\vdash \Delta, A^\perp}{\vdash \Delta, ?^i A^\perp}}{\vdash \Gamma, \Delta} \quad \rightsquigarrow \quad \dfrac{\vdash \Gamma, A \quad \vdash \Delta, A^\perp}{\vdash \Gamma, \Delta}$$

In the next transformation, the need for the restrictions imposed on subexponential signatures Σ becomes apparent. Because of the application of the promotion rule for $!^i$ in the derivation on the left, it must be the case that all formulas in Γ have as main connective a subexponential question-mark with index greater or equal to i. Moreover, because of the weakening of the formula $?^i A^\perp$, it must be the case that $i \in \mathcal{W}$, and hence, from the restrictions on consistent subexponential signatures, all formulas in Γ can also be weakened. This allows us to use the derivation on the right with lower degree.

$$\dfrac{\dfrac{\vdash \Gamma, A}{\vdash \Gamma, !^i A} \quad \dfrac{\vdash \Delta}{\vdash \Delta, ?^i A^\perp}}{\vdash \Gamma, \Delta} \quad \rightsquigarrow \quad \dfrac{\vdash \Delta}{\vdash \Gamma, \Delta}$$

There is another transformation, called pseudo key-case (shown below), which involves contraction and is necessary for the cut-elimination algorithm, but that does not decrease the degree of proofs. This case highlights the need for the restriction that subexponentials indexes are upwardly closed with respect to contraction. Here, since the promotion rule is applicable in the left branch of the derivation to the left, the main connective of the formulas in Γ must be subexponential question-mark with index greater or equal to i, and, therefore, one can also contract Γ because $i \in \mathcal{C}$.

$$\dfrac{\dfrac{\vdash \Gamma, A}{\vdash \Gamma, !^i A} \quad \dfrac{\vdash \Delta, ?^i A^\perp, ?^i A^\perp}{\vdash \Delta, ?^i A^\perp}}{\vdash \Gamma, \Delta} \quad \rightsquigarrow \quad \dfrac{\dfrac{\vdash \Gamma, A}{\vdash \Gamma, !^i A} \quad \dfrac{\vdash \Gamma, A}{\vdash \Gamma, !^i A} \quad \vdash \Delta, ?^i A^\perp, ?^i A^\perp}{\dfrac{\vdash \Gamma, \Gamma, \Delta, ?^i A^\perp}{\vdash \Gamma, \Gamma, \Delta}}}{\vdash \Gamma, \Delta}$$

Then the proof proceeds as in the proof of cut-elimination in linear logic, by induction on the height of trees and using the key-cases and pseudo key-cases to eliminate cuts in the proof. For example, if the proof is of the form to the left, where $(F)^{n-1}$ denotes that there are $n-1$ copies of the formula F, then we can permute the dereliction downwards, obtaining the derivation to the right:

$$\dfrac{\dfrac{\vdash \Gamma, A}{\vdash \Gamma, !^i A} \quad \dfrac{\dfrac{\vdash \Delta, (?^i A^\perp)^{n-1}, A^\perp}{\vdash \Delta, (?^i A^\perp)^{n-1}, ?^i A^\perp} \quad \vdash \Delta, ?^i A^\perp}{\vdash \Gamma, \Delta} \quad \rightsquigarrow \quad \dfrac{\vdash \Gamma, A}{\vdash \Gamma, !^i A} \quad \dfrac{\dfrac{\vdash \Delta, (?^i A^\perp)^{n-1}, A^\perp}{\vdash \Delta, ?^i A^\perp, A^\perp} \quad \vdash \Delta, ?^i A^\perp}{\vdash \Gamma, \Delta}$$

We then apply the pseudo key-case transformation and permute the cut upwards, obtaining the derivation to the left and to the right respectively:

$$\dfrac{\dfrac{\vdash \Gamma, A}{\vdash \Gamma, !^i A} \quad \dfrac{\vdash \Gamma, A}{\vdash \Gamma, !^i A} \quad \dfrac{\vdash \Delta, (?^i A^\perp)^{n-1}, A^\perp}{\vdash \Delta, ?^i A^\perp, A^\perp} \quad \vdash \Delta, ?^i A^\perp, ?^i A^\perp}{\dfrac{\vdash \Gamma, \Gamma, \Delta, ?^i A^\perp}{\vdash \Gamma, \Gamma, \Delta}}}{\vdash \Gamma, \Delta} \quad \rightsquigarrow \quad \dfrac{\dfrac{\vdash \Gamma, A}{\vdash \Gamma, !^i A} \quad \dfrac{\vdash \Gamma, A}{\vdash \Gamma, !^i A} \quad \dfrac{\dfrac{\vdash \Delta, (?^i A^\perp)^{n-1}, A^\perp}{\vdash \Delta, ?^i A^\perp, A^\perp} \quad \vdash \Delta, ?^i A^\perp, A^\perp}{\vdash \Gamma, \Delta, A^\perp} \quad \vdash \Delta, ?^i A^\perp}{\dfrac{\vdash \Gamma, \Gamma, \Delta, ?^i A^\perp}{\vdash \Gamma, \Gamma, \Delta}}}{\vdash \Gamma, \Delta}$$

Now, we can use the key-case to obtain the following derivation, on which we apply the induction hypothesis to the adequate subtree, to obtain a proof of smaller degree:

$$\dfrac{\dfrac{\dfrac{\vdash \Gamma, A}{\vdash \Gamma, !^i A} \quad \dfrac{\vdash \Delta, (?^i A^\perp)^{n-1}, A^\perp}{\vdash \Delta, ?^i A^\perp, A^\perp}}{\vdash \Gamma, A \quad \vdash \Gamma, \Delta, A^\perp}}{\dfrac{\vdash \Gamma, \Gamma, \Delta}{\vdash \Gamma, \Delta}}$$

Here, we do not show the rest of the proof, but we invite the more interested reader to have a look at the technical report [Braüner 1996]. □

5.2 Focusing in linear logic with subexponentials

To propose a focused system for $SELL_\Sigma$, we follow the steps of Andreoli [Andreoli 1992] and of Saurin *et al.* [Miller 2007b] by first proposing a proof system that incorporates the structural rules

for contraction and weakening into the logical rules. However, since we are dealing with more than two modalities, we propose, instead of a *dyadic* system, a *polyadic* system where there is one different context for each subexponential index.

As proof search is performed from the root towards its leaves, one wants to postpone as much as possible the application of structural rules. In the dyadic system proposed by Andreoli, this corresponds to applying weakening rules just before initial rules and contractions before the tensor and dereliction rules. We will use the same motivation when proposing the polyadic system, but some extra considerations arise when considering subexponentials.

Differently from linear logic, the weakening rule does not permute with all logical rules, namely it does not permute over promotion rules. Consider for example the sequent $\vdash ?^a P, !^b Q$. Because of the side condition in the promotion rule, the weakening of $?^a P$ (here assuming that $a \in \mathcal{W}$) will permute over the promotion rule if $b \preceq a$, otherwise one must first weaken $?^a P$. Hence, the *polyadic* system for such a logic differs from Andreoli's *dyadic* system for linear logic, as weakening occurs not only before initial rules, but also immediately before promotion rules.

Contraction is also a bit more involved than in Andreoli's dyadic system. Because of relevant formulas, that is, formulas that can contract but not weaken, we cannot freely perform contraction of this type of formulas since once contracted the new formulas created must later be used. On the other hand, one can contract unbounded formulas without losing completeness because, once contracted, the created formulas can be later weakened. In the next subsection, we start with the simpler case for proof systems constructed using subexponential signatures, $\langle \mathcal{I}, \preceq, \mathcal{W}, \mathcal{C} \rangle$, such that $\mathcal{C} \subseteq \mathcal{W}$, that is, proof systems that do not allow relevant formulas. Later, we will discuss the case when relevant formulas are also permitted.

5.2.1 Case without relevant formulas

A polyadic system is constructed by using polyadic sequents. Polyadic sequents are specified as follows:

Definition 5.4 Let $\Sigma = \langle \mathcal{I}, \preceq, \mathcal{W}, \mathcal{C} \rangle$ be a subexponential signature. Then a polyadic sequent for Σ is a pair, written as $\vdash \mathcal{K} : \Gamma$, with a multiset of formulas, Γ, and with a function, $\mathcal{K}$, from the set of subexponential indexes $\mathcal{I}$ to the set of multisets of formulas.

The polyadic sequent $\vdash \mathcal{K} : \Gamma$ denotes the linear logic sequent, $\vdash ?^{l_1}\mathcal{K}[l_1], \ldots, ?^{l_n}\mathcal{K}[l_n], \Gamma$ obtained by collecting the formulas in the multiset $\mathcal{K}[l_i]$, for each l_i in the domain of $\mathcal{K}$, and prefixing all the formulas in $\mathcal{K}[l_i]$ with the subexponential $?^{l_i}$. Moreover, we derive the function of the polyadic sequent corresponding to a monadic sequent as follows: let $\mathcal{S}$ be the monadic sequent $\vdash ?^{x_1}\Theta_1, \ldots, ?^{x_n}\Theta_n, \Gamma$, then the function $\mathcal{K}_\mathcal{S}$ of a corresponding polyadic sequent $\vdash \mathcal{K}_\mathcal{S} : \Gamma$ is defined as follows:

$$\mathcal{K}_\mathcal{S}[c] = \begin{cases} \Theta_i & \text{if } c = x_i \\ \emptyset & \text{if } c \notin \{x_1, \ldots, x_n\} \end{cases}$$

This is a straightforward generalization of Andreoli's dyadic sequent. Notice that Γ might contain formulas whose main connective is a subexponential question-mark, and, therefore, the same monadic sequent might have several associated polyadic sequents. However, we will make sure that it is clear from the context which multisets Θ_i in a sequent $\mathcal{S}$ that we are using to construct the function $\mathcal{K}_\mathcal{S}$.

Before we show the polyadic proof system for *SELL*, we first specify some operations over functions of polyadic sequents $\mathcal{K}, \mathcal{K}_1$ and $\mathcal{K}_2$ for a subexponential signature $\Sigma = \langle \mathcal{I}, \preceq, \mathcal{W}, \mathcal{C} \rangle$:

- $\mathcal{K} \leq_i [l] = \begin{cases} \mathcal{K}[l] & \text{if } i \preceq l \\ \emptyset & \text{if } i \not\preceq l \end{cases}$

  where $i \in \mathcal{I}$ is a subexponential index.

- $\mathcal{K}[\mathcal{J}] = \bigcup\{\mathcal{K}[i] \mid i \in \mathcal{J}\}$
  where $\mathcal{J} \subseteq \mathcal{I}$ is a set of subexponential indexes.
- $(\mathcal{K} +_l A)[i] = \begin{cases} \mathcal{K}[i] \cup \{A\} & \text{if } i = l \\ \mathcal{K}[i] & \text{otherwise} \end{cases}$
  where A is a formula.
- Let $\mathcal{J} \subseteq \mathcal{I}$ be a set of subexponential indexes, and $\star \in \{=, \subset, \subseteq\}$ be a binary connective. Then $(\mathcal{K}_1 \star \mathcal{K}_2) \mid_{\mathcal{J}}$ is true if and only if $\forall i \in \mathcal{J}.(\mathcal{K}_1[i] \star \mathcal{K}_2[i])$.
- $(\mathcal{K}_1 \otimes \mathcal{K}_2)[i] = \begin{cases} \mathcal{K}_1[i] \cup \mathcal{K}_2[i] & \text{if } i \notin \mathcal{C} \\ \mathcal{K}_1[i] & \text{otherwise if } i \in \mathcal{C} \cap \mathcal{W} \end{cases}$
  Here, we use the multiset union.

Notice that the function $K_1 \otimes K_2$ is only well defined when $\mathcal{C} \subseteq \mathcal{W}$, which is assumed true in this subsection. In the next subsection, we will fill in the gap.

The rules for the polyadic system, $SELL^p_\Sigma$ for $SELL_\Sigma$, are given in Figure 5.1. The promotion rule and the one introduction rules are the most interesting ones. Their side-conditions enforces that the only side-formulas are those that allows one to introduce the main connective as seen in the previous section. For example, in the promotion rule, one must make sure that the side-formulas premise only contains formulas whose main connective is a question-mark with an index greater or equal to the index of the bang introduced.

Theorem 5.5 *Let $\Sigma = \langle \mathcal{I}, \preceq, \mathcal{W}, \mathcal{C}\rangle$ be a subexponential signature, such that $\{x_1, \ldots, x_n\} \subseteq \mathcal{I}$ and $\mathcal{C} \subseteq \mathcal{W}$, and let $\mathcal{S}$ be the sequent $\vdash ?^{x_1}\Theta_1, \ldots, ?^{x_n}\Theta_n, \Gamma$. Then,*

$$\mathcal{S} \text{ is provable in } SELL_\Sigma \text{ iff } \vdash \mathcal{K}_\mathcal{S} : \Gamma \text{ is provable in } SELL^p_\Sigma$$

Proof We make use of the following lemma, which is proved by induction on the height of derivations.

Lemma 5.6 *Let $\Sigma = \langle \mathcal{I}, \preceq, \mathcal{W}, \mathcal{C}\rangle$ be a subexponential signature, such that $\{y_1, \ldots, y_n\} \subseteq \mathcal{W}$, and let $\Theta_1, \ldots, \Theta_n$ be multisets of formulas. If $\vdash \mathcal{K}_1 : \Gamma$ has a proof in $SELL^p_\Sigma$ then $\vdash \mathcal{K}_2 : \Gamma$ has a proof of the same height in $SELL^p_\Sigma$, where $\mathcal{K}_2[y_i] = \mathcal{K}_1[y_i] \cup \{\Theta_i\}$, for all $i = 1, \ldots, n$, and $\mathcal{K}_2[c] = \mathcal{K}_1[c]$ for all $c \notin \{y_1, \ldots, y_n\}$.*

Proof By induction on the height of proofs. □

We prove the completeness direction by induction on the height of trees. Here we show some of the inductive cases, according to the last rule in the proof of $\vdash ?^{x_1}\Theta_1, \ldots, ?^{x_n}\Theta_n, \Gamma$.

- Case $\parr$:

$$\frac{\vdash ?^{x_1}\Theta_1, \ldots, ?^{x_n}\Theta_n, \Gamma, A, B}{\vdash ?^{x_1}\Theta_1, \ldots, ?^{x_n}\Theta_n, \Gamma, A \parr B}\ [\parr]$$

If the premise of this rule is called $\mathcal{S}_1$ and its conclusion $\mathcal{S}$, then, from the induction hypothesis, there is a proof of $\vdash \mathcal{K}_{\mathcal{S}_1} : \Gamma, A, B$ in $SELL^p_\Sigma$. It is easy to check from their definitions that the functions $\mathcal{K}_{\mathcal{S}_1}$ and $\mathcal{K}_\mathcal{S}$ are equivalent. Therefore, from the $\parr$ introduction rule in $SELL^p_\Sigma$, there is also proof of the sequent $\vdash \mathcal{K}_\mathcal{S} : \Gamma, A \parr B$ in $SELL^p_\Sigma$.

The cases for the rules $\top, \bot, \&, \oplus_i, 1, \exists$, and $\forall$ are similar.

- Case $\otimes$:

$$\frac{\vdash ?^{x_1}\Theta^1_1, \ldots, ?^{x_n}\Theta^1_n, \Gamma, A \qquad \vdash ?^{x_1}\Theta^2_1, \ldots, ?^{x_n}\Theta^2_n, \Delta, B}{\vdash ?^{x_1}\Theta^1_1, ?^{x_1}\Theta^2_1, \ldots, ?^{x_n}\Theta^1_n, ?^{x_n}\Theta^2_n, \Gamma, \Delta, A \otimes B}\ [\otimes]$$

Assume that the premises of this rule are called $\mathcal{S}_1$ and $\mathcal{S}_2$ and its conclusion, $\mathcal{S}$, then, by induction hypothesis, there are proofs in $SELL^p_\Sigma$ of the sequents $\vdash \mathcal{K}_{\mathcal{S}_1}, \Gamma, A$ and $\vdash \mathcal{K}_{\mathcal{S}_2}, \Delta, B$. From Lemma 5.6, there are proofs of the same height for the sequents $\vdash \mathcal{K}'_{\mathcal{S}_1}, \Gamma, A$ and $\vdash \mathcal{K}'_{\mathcal{S}_2}, \Delta, B$, but where $\mathcal{K}'_{\mathcal{S}_i}[x_i] = \Theta^1_i \cup \Theta^2_i$ for all $x_i \in \mathcal{C} \cap \mathcal{W}$ and $\mathcal{K}'_{\mathcal{S}_i}[c] = \mathcal{K}_{\mathcal{S}_i}[c]$ for all other indexes c. Hence, there is also a proof of the sequent $\vdash \mathcal{K}_{\mathcal{S}}, \Gamma, \Delta, A \otimes B$.

- Case $D?^{x_i}$: there are two subcases to be considered:

(1) If the dereliction occurs in a formula in Θ_i:

$$\frac{\vdash ?^{x_1}\Theta_1, \ldots, ?^{x_1}\Theta_i, \ldots ?^{x_n}\Theta_n, \Gamma, P}{\vdash ?^{x_1}\Theta_1, \ldots, ?^{x_1}\Theta_i, ?^{x_i}P, \ldots ?^{x_n}\Theta_n, \Gamma}\ [D?^{x_i}]$$

Assume that the premise of the rule is called $\mathcal{S}_1$ and its conclusion, $\mathcal{S}$, then by the induction hypothesis there is a proof of $\vdash \mathcal{K}_{\mathcal{S}_1} : \Gamma, P$ in $SELL^p_\Sigma$. Now, we distinguish two subcases: a) if $x_i \notin \mathcal{C} \cap \mathcal{W}$, then, from the rule $D_2?^{x_i}$, the sequent $\vdash \mathcal{K}_{\mathcal{S}} : \Gamma$ has a proof in $SELL^p_\Sigma$; b) if $x_i \in \mathcal{C} \cap \mathcal{W}$, then, from the Lemma 5.6, the sequent $\vdash \mathcal{K}_{\mathcal{S}} : \Gamma, P$ has a proof in $SELL^p_\Sigma$ of the same height. Finally, from the rule $D_1?^{x_i}$, the sequent $\vdash \mathcal{K}_{\mathcal{S}} : \Gamma$ is provable in $SELL^p_\Sigma$.

(2) If the dereliction occurs in a formula in Γ:

$$\frac{\vdash ?^{x_1}\Theta_1, \ldots, ?^{x_n}\Theta_n, \Gamma, P}{\vdash ?^{x_1}\Theta_1, \ldots, ?^{x_n}\Theta_n, \Gamma, ?^c P}\ [D?^{x_i}]$$

In this case the reasoning is similar to the previous case, only that we first apply an instance of the rule $?^c$ rule:

$$\frac{\vdash \mathcal{K}_{\mathcal{S}} +_c P : \Gamma}{\vdash \mathcal{K}_{\mathcal{S}} : \Gamma, ?^c P}\ [?^c]$$

The cases for the rule $C?^c, W?^c$ and $!^c$ are again similar. One just needs to use the fact that contractions permutes over all rules except tensor and dereliction rules and that weakening permutes over all rules except promotion rules.

Now we prove the soundness direction, which is also by induction on the height of proofs. Here we show only some of the cases:

$$\frac{}{\vdash \mathcal{K} : A^\perp, A}\ [I] \qquad \rightsquigarrow \qquad \frac{\dfrac{}{\vdash A^\perp, A}\ [I]}{\vdash ?^{x_1}\mathcal{K}[x_1], \ldots, ?^{x_n}\mathcal{K}[x_n], A^\perp, A}\ [m \times W?^{x_i}]$$

Notice that, only because of the side condition $\mathcal{K}[\mathcal{I} \setminus \mathcal{W}] = \emptyset$ in $SELL^p_\Sigma$'s initial rule, we can weaken all the formulas in the context.

The case for the rule 1 is similar.

$$\frac{\vdash \mathcal{K} : \Gamma, A, B}{\vdash \mathcal{K} : \Gamma, A \parr B}\ [\parr] \qquad \rightsquigarrow \qquad \frac{\vdash ?^{x_1}\mathcal{K}[x_1], \ldots, ?^{x_n}\mathcal{K}[x_n], \Gamma, A, B}{\vdash ?^{x_1}\mathcal{K}[x_1], \ldots, ?^{x_n}\mathcal{K}[x_n], \Gamma, A \parr B}\ [\parr]$$

The cases for $\&, \forall, \exists, \top, \bot$, and $\oplus_i$ are similar.

$$\frac{\vdash \mathcal{K}_1 : \Gamma, A \quad \vdash \mathcal{K}_2 : \Delta, B}{\vdash \mathcal{K}_1 \otimes \mathcal{K}_2 : \Gamma, \Delta, A \otimes B}\ [\otimes] \qquad \rightsquigarrow \qquad \frac{\dfrac{\vdash \Lambda_u, \Lambda^1_b, \Gamma, A \quad \vdash \Lambda_u, \Lambda^2_b, \Delta, B}{\vdash \Lambda_u, \Lambda_u, \Lambda^1_b, \Lambda^2_b, \Gamma, \Delta, A \otimes B}\ [\otimes]}{\vdash \Lambda_u, \Lambda^1_b, \Lambda^2_b, \Gamma, \Delta, A \otimes B}\ [m \times C?^{x_i}]$$

where $\Lambda_u = \{?^{x_i}P \mid x_i \in \mathcal{C} \text{ and } P \in (\mathcal{K}_1 \otimes \mathcal{K}_2)[x_i]\}$ and $\Lambda^j_b = \{?^{x_i}P \mid x_i \notin \mathcal{C} \text{ and } P \in \mathcal{K}_j[x_i]\}$, for $j = 1, 2$.

$$\frac{\vdash \mathcal{K} \leq_l : A}{\vdash \mathcal{K} : !^l A}\,[!^l] \qquad \rightsquigarrow \qquad \cfrac{\cfrac{\vdash \Lambda_{\geq}, A}{\vdash \Lambda_{\geq}, !^l A}\,[!^l]}{\vdash \Lambda_{\geq}, \Lambda_{\not\geq}, !^l A}\,[m \times W ?^{x_i}]$$

where $\Lambda_{\geq} = \{?^{x_i} P \mid l \preceq x_i \text{ and } P \in \mathcal{K}[x_i]\}$ and $\Lambda_{\not\geq} = \{?^{x_i} P \mid l \not\preceq x_i \text{ and } P \in \mathcal{K}[x_i]\}$. Notice that we can weaken the set $\Lambda_{\not\geq}$ because of the side condition $\mathcal{K}[\{x \mid l \not\preceq x \wedge x \notin \mathcal{W}\}] = \emptyset$ in $SELL^p_\Sigma$'s promotion rule.

$$\frac{\vdash \mathcal{K} : A, \Gamma}{\vdash \mathcal{K} : \Gamma}\,[D_1 ?^{x_i}] \qquad \rightsquigarrow \qquad \cfrac{\cfrac{\vdash ?^{x_1}\Theta_1, \ldots, ?^{x_1}\Theta_i, ?^{x_i} A, \ldots ?^{x_n}\Theta_n, \Gamma, A}{\vdash ?^{x_1}\Theta_1, \ldots, ?^{x_1}\Theta_i, ?^{x_i} A, ?^{x_i} A, \ldots ?^{x_n}\Theta_n, \Gamma}\,[D ?^{x_i}]}{\vdash ?^{x_1}\Theta_1, \ldots, ?^{x_1}\Theta_i, ?^{x_i} A, \ldots ?^{x_n}\Theta_n, \Gamma}\,[C ?^{x_i}]$$

Notice that we can contract the formula $?^{x_i} A$ because of $D_1 ?^c$'s side condition $A \in \mathcal{K}[\mathcal{C} \cap \mathcal{W}]$.

$$\frac{\vdash \mathcal{K} : A, \Gamma}{\vdash \mathcal{K} +_l A : \Gamma}\,[D_2 ?^l] \qquad \rightsquigarrow \qquad \frac{\vdash ?^{x_1}\Theta_1, \ldots ?^{x_n}\Theta_n, A, \Gamma}{\vdash ?^{x_1}\Theta_1, \ldots ?^{x_n}\Theta_n, ?^l A, \Gamma}\,[D ?^l]$$

□

Now we are ready to introduce the focused system $SELLF_\Sigma$, depicted in Figure 5.2. As in Andreoli's *tryadic* system, we add the context to the left of the arrows $\Uparrow$ and $\Downarrow$ that contains only positive formulas and literals. Notice that one could incorporate this new context in the function $\mathcal{K}$ by assigning a new element in its domain, say $-\infty$, such that $\mathcal{K}[-\infty]$ returns the multiset of formulas in this context. However, we do not have any good reason to do so, and, therefore, we prefer the system closer to Andreoli's tryadic system.

For proving the completeness of $SELLF_\Sigma$, we adapt the machinery introduced in the proof of the focusing theorem for linear logic in Chapter 2 (see Theorem 2.8). Notice that we assume here a global polarity assignment for literals. However, one could easily adapt the proof to accommodate more flexible polarity assignments such as the ones proposed by Saurin *et al.* in [Miller 2007b].

Definition 5.7 Let $\Sigma = \langle \mathcal{I}, \preceq, \mathcal{W}, \mathcal{C} \rangle$ be a subexponential signature. For all $l \in \mathcal{I}$, the subexponential connectives $!^l$ are classified as synchronous and the subexponential connectives $?^l$ as asynchronous.

Lemma 5.8 *Let α be an inference rule and β be an asynchronous rule. Then α/β.*

Proof The permutation cases are similar to those in Lemma 2.10. For example, the case for $\otimes/\&$ is shown below:

$$\cfrac{\cfrac{\vdash \mathcal{K}_1 : \Gamma, F, A \quad \vdash \mathcal{K}_1 : \Gamma, F, B}{\vdash \mathcal{K}_1 : \Gamma, F, A \,\&\, B}\,[\&] \quad \vdash \mathcal{K}_2 : \Delta, G}{\vdash \mathcal{K}_1 \otimes \mathcal{K}_2 : \Gamma, \Delta, F \otimes G, A \,\&\, B}\,[\otimes]$$

the tensor rule permutes over the with rule as follows:

$$\cfrac{\cfrac{\vdash \mathcal{K}_1 : \Gamma, F, A \quad \vdash \mathcal{K}_2 : \Delta, G}{\vdash \mathcal{K}_1 \otimes \mathcal{K}_2 : \Gamma, \Delta, F \otimes G, A}\,[\otimes] \quad \cfrac{\vdash \mathcal{K}_1 : \Gamma, F, B \quad \vdash \mathcal{K}_2 : \Delta, G}{\vdash \mathcal{K}_1 \otimes \mathcal{K}_2 : \Gamma, \Delta, F \otimes G, B}\,[\otimes]}{\vdash \mathcal{K}_1 \otimes \mathcal{K}_2 : \Gamma, \Delta, F \otimes G, A \,\&\, B}\,[\&]$$

The subexponentials do not cause problems. For example, the $\oplus_i$ rules permute over $?^l$ rules:

$$\cfrac{\cfrac{\mathcal{K} +_l P : \Gamma, F}{\mathcal{K} : \Gamma, F, ?^l P}\,[?^l]}{\mathcal{K} : \Gamma, F \oplus G, ?^l P}\,[\oplus_1] \qquad \rightsquigarrow \qquad \cfrac{\cfrac{\mathcal{K} +_l P : \Gamma, F}{\mathcal{K} +_l P : \Gamma, F \oplus G}\,[\oplus_1]}{\mathcal{K} : \Gamma, F \oplus G, ?^l P}\,[?^l]$$

□

Logical Rules

$$\frac{\vdash \mathcal{K} : \Gamma, A \quad \vdash \mathcal{K} : \Gamma, B}{\vdash \mathcal{K} : \Gamma, A \,\&\, B}\ [\&] \qquad \frac{\vdash \mathcal{K} : \Gamma, A, B}{\vdash \mathcal{K} : \Gamma, A \parr B}\ [\parr]$$

$$\frac{}{\vdash \mathcal{K} : \Gamma, \top}\ [\top] \qquad \frac{\vdash \mathcal{K} : \Gamma}{\vdash \mathcal{K} : \Gamma, \bot}\ [\bot]$$

$$\frac{\vdash \mathcal{K} : \Gamma, A\{t/x\}}{\vdash \mathcal{K} : \Gamma, \exists x\, A}\ [\exists] \qquad \frac{\vdash \mathcal{K} : \Gamma, A\{c/x\}}{\vdash \mathcal{K} : \Gamma, \forall x\, A}\ [\forall]$$

$$\frac{\vdash \mathcal{K} : \Gamma, A_i}{\vdash \mathcal{K} : \Gamma, A_1 \oplus A_2}\ [\oplus_i] \qquad \frac{\vdash \mathcal{K}_1 : \Gamma, A \quad \vdash \mathcal{K}_2 : \Delta, B}{\vdash \mathcal{K}_1 \otimes \mathcal{K}_2 : \Gamma, \Delta, A \otimes B}\ [\otimes,\ \text{provided that } (\mathcal{K}_1 = \mathcal{K}_2)\,|_{\mathcal{C}\cap\mathcal{W}}]$$

$$\frac{}{\vdash \mathcal{K} : 1}\ [1,\ \text{provided } \mathcal{K}[\mathcal{I} \setminus \mathcal{W}] = \emptyset]$$

$$\frac{\vdash \mathcal{K} \leq_l : A}{\vdash \mathcal{K} : !^l\, A}\ [!^l,\ \text{provided } \mathcal{K}[\{x \mid l \not\preceq x \wedge x \notin \mathcal{W}\}] = \emptyset]$$

$$\frac{}{\vdash \mathcal{K} : A^\perp, A}\ [I,\ \text{provided } \mathcal{K}[\mathcal{I} \setminus \mathcal{W}] = \emptyset]$$

$$\frac{\vdash \mathcal{K} : A, \Gamma}{\vdash \mathcal{K} : \Gamma}\ [D_1?^l,\ \text{provided } A \in \mathcal{K}[\mathcal{C} \cap \mathcal{W}]] \qquad \frac{\vdash \mathcal{K} : A, \Gamma}{\vdash \mathcal{K} +_l A : \Gamma}\ [D_2?^l,\ \text{provided } l \notin \mathcal{C} \cap \mathcal{W}]$$

Structural Rules

$$\frac{\vdash \mathcal{K} +_l A : \Gamma}{\vdash \mathcal{K} : \Gamma, ?^l A}\ [?^l]$$

Figure 5.1: The polyadic system $SELL^p_\Sigma$.

Asynchronous Phase

$$\frac{\vdash \mathcal{K} : \Gamma \Uparrow L, A \quad \vdash \mathcal{K} : \Gamma \Uparrow L, B}{\vdash \mathcal{K} : \Gamma \Uparrow L, A \,\&\, B}\ [\&] \qquad \frac{\vdash \mathcal{K} : \Gamma \Uparrow L, A, B}{\vdash \mathcal{K} : \Gamma \Uparrow L, A \parr B}\ [\parr]$$

$$\frac{}{\vdash \mathcal{K} : \Gamma \Uparrow L, \top}\ [\top] \qquad \frac{\vdash \mathcal{K} : \Gamma \Uparrow L}{\vdash \mathcal{K} : \Gamma \Uparrow L, \bot}\ [\bot]$$

$$\frac{\vdash \mathcal{K} : \Gamma \Uparrow L, A\{c/x\}}{\vdash \mathcal{K} : \Gamma \Uparrow L, \forall x\, A}\ [\forall] \qquad \frac{\vdash \mathcal{K} +_l A : \Gamma \Uparrow L}{\vdash \mathcal{K} : \Gamma \Uparrow L, ?^l A}\ [?^l]$$

Synchronous Phase

$$\frac{\vdash \mathcal{K} : \Gamma \Downarrow A_i}{\vdash \mathcal{K} : \Gamma \Downarrow A_1 \oplus A_2}\ [\oplus_i] \qquad \frac{\vdash \mathcal{K}_1 : \Gamma \Downarrow A \quad \vdash \mathcal{K}_2 : \Delta \Downarrow B}{\vdash \mathcal{K}_1 \otimes \mathcal{K}_2 : \Gamma, \Delta \Downarrow A \otimes B}\ [\otimes, \text{ provided } (\mathcal{K}_1 = \mathcal{K}_2)\,|_{\mathcal{C}\cap\mathcal{W}}]$$

$$\frac{}{\vdash \mathcal{K} : \cdot \Downarrow 1}\ [1, \text{ provided } \mathcal{K}[\mathcal{I} \setminus \mathcal{W}] = \emptyset] \qquad \frac{\vdash \mathcal{K} : \Gamma \Downarrow A\{t/x\}}{\vdash \mathcal{K} : \Gamma \Downarrow \exists x\, A}\ [\exists]$$

$$\frac{\vdash \mathcal{K} \leq_l : \cdot \Uparrow A}{\vdash \mathcal{K} : \cdot \Downarrow !^l A}\ [!^l, \text{ provided } \mathcal{K}[\{x \mid l \not\preceq x \wedge x \notin \mathcal{W}\}] = \emptyset]$$

Reaction, Identity, and Decide Rules

$$\frac{}{\vdash \mathcal{K} : \Gamma \Downarrow A_p}\ [I, \text{ provided } A_p^\perp \in (\Gamma \cup \mathcal{K}[\mathcal{I}]) \text{ and } (\Gamma \cup \mathcal{K}[\mathcal{I} \setminus \mathcal{W}]) \subseteq \{A_p^\perp\}]$$

$$\frac{\vdash \mathcal{K} +_l P : \Gamma \Downarrow P}{\vdash \mathcal{K} +_l P : \Gamma \Uparrow \cdot}\ [D_l, \text{ provided } l \in \mathcal{C} \cap \mathcal{W}] \qquad \frac{\vdash \mathcal{K} : \Gamma \Downarrow P}{\vdash \mathcal{K} +_l P : \Gamma \Uparrow \cdot}\ [D_l, \text{ provided } l \notin \mathcal{C} \cap \mathcal{W}]$$

$$\frac{\vdash \mathcal{K} : \Gamma \Downarrow P}{\vdash \mathcal{K} : \Gamma, P \Uparrow \cdot}\ [D_1] \qquad \frac{\vdash \mathcal{K} : \Gamma \Uparrow N}{\vdash \mathcal{K} : \Gamma \Downarrow N}\ [R\Downarrow] \qquad \frac{\vdash \mathcal{K} : \Gamma, S \Uparrow L}{\vdash \mathcal{K} : \Gamma \Uparrow L, S}\ [R\Uparrow]$$

Figure 5.2: The focused system $SELLF_\Sigma$. Here, $\Sigma = \langle \mathcal{I}, \preceq, \mathcal{W}, \mathcal{C} \rangle$ is a subexponential signature, such that $\mathcal{C} \subseteq \mathcal{W}$; L is a list of formulas; Γ is a multiset of positive formulas and literals; A_p is a positive polarity literal; P is not a negative polarity literal; S is a positive formula or a literal; N is a negative formula.

Lemma 5.9 *Let α and β be two synchronous rules. Then α/β.*

Proof The proof is similar to the Lemma 2.11. We show only some of the cases:

- $(\otimes/\oplus_1)$:

$$\dfrac{\dfrac{\vdash \mathcal{K}_1 : \Gamma, F, P}{\vdash \mathcal{K}_1 : \Gamma, F, P \oplus Q}\,[\oplus_1] \quad \vdash \mathcal{K}_2 : \Delta, G}{\vdash \mathcal{K}_1 \otimes \mathcal{K}_2 : \Gamma, \Delta, F \otimes G, P \oplus Q}\,[\otimes] \quad \rightsquigarrow \quad \dfrac{\vdash \mathcal{K}_1 : \Gamma, F, P \quad \vdash \mathcal{K}_2 : \Delta, G}{\dfrac{\vdash \mathcal{K}_1 \otimes \mathcal{K}_2 : \Gamma, \Delta, F \otimes G, P}{\vdash \mathcal{K}_1 \otimes \mathcal{K}_2 : \Gamma, \Delta, F \otimes G, P \oplus Q}\,[\oplus_1]}\,[\otimes]$$

- $(\oplus_1/\oplus_2)$:

$$\dfrac{\dfrac{\vdash \mathcal{K} : \Gamma, F, Q}{\vdash \mathcal{K} : \Gamma, F, P \oplus Q}\,[\oplus_2]}{\vdash \mathcal{K} : \Gamma, F \oplus G, P \oplus Q}\,[\oplus_1] \quad \rightsquigarrow \quad \dfrac{\dfrac{\vdash \mathcal{K} : \Gamma, F, Q}{\vdash \mathcal{K} : \Gamma, F \oplus G, Q}\,[\oplus_1]}{\vdash \mathcal{K} : \Gamma, F \oplus G, P \oplus Q}\,[\oplus_2]$$

- $(\otimes/\exists)$:

$$\dfrac{\dfrac{\vdash \mathcal{K}_1 : \Gamma, F, P[t/x]}{\vdash \mathcal{K}_1 : \Gamma, F, \exists x P}\,[\exists] \quad \vdash \mathcal{K}_2 : \Delta, G}{\vdash \mathcal{K}_1 \otimes \mathcal{K}_2 : \Gamma, \Delta, F \otimes G, \exists x P}\,[\otimes] \quad \rightsquigarrow \quad \dfrac{\vdash \mathcal{K}_1 : \Gamma, F, P[t/x] \quad \vdash \mathcal{K}_2 : \Delta, G}{\dfrac{\vdash \mathcal{K}_1 \otimes \mathcal{K}_2 : \Gamma, \Delta, F \otimes G, P[t/x]}{\vdash \mathcal{K}_1 \otimes \mathcal{K}_2 : \Gamma, \Delta, F \otimes G, \exists x P}\,[\exists]}\,[\otimes]$$

□

We use the same definition of positive trunks given in Definition 2.13. We also distinguish each occurrence of formulas, F, created from dereliction rules, by assigning to it a different number i, like (F, i), as specifies the following definition.

Definition 5.10 Given a positive trunk, Π, of the sequent $\vdash \mathcal{K} : \Gamma$, we assign to every occurrence of a dereliction rule, $D_1?^l$ and $D_2?^l$, in Π, a unique index (F, i) to the occurrence of formula F created. The active formulas in Π are the active formulas in Γ and the indexed formulas (F, i).

We also use the same relation $\prec_f$ specified in Definition 2.15 and show that this relation is still acyclic.

Lemma 5.11 *The relation $\prec_f$ is acyclic.*

Proof The proof is similar to the proof of Lemma 2.16. The new promotion and dereliction rules do not cause any problems for the same reasons as in linear logic. □

Lemma 5.12 *Let Π be a positive trunk, $\mathcal{S}$ be the root sequent of Π such that it cannot be the conclusion of any asynchronous rule, and F be a minimal element in Π. If F is a formula created by a dereliction, then there is a proof of $\mathcal{S}$ where F is created and a rule is applied to it last. Otherwise, if F is not created by a dereliction, then there is a proof where a rule is applied to F last.*

Proof The proof is similar to the proof of Lemma 2.18. □

The completeness of $SELLF_\Sigma$ follows with the same reasoning as Theorem 2.8 by using the permutations lemmas and the focalisation graph above.

Theorem 5.13 *Let $\Sigma = \langle \mathcal{I}, \preceq, \mathcal{W}, \mathcal{C} \rangle$ be a subexponential signature, such that $\mathcal{C} \subseteq \mathcal{W}$. Then $SELLF_\Sigma$ is sound and complete with respect to $SELL_\Sigma$.*

5.2.2 Adding relevant formulas

In the previous subsection, we assumed that $\mathcal{C} \subseteq \mathcal{W}$ and, therefore, performing contraction did not affect provability because, after creating a formula by using the contraction rule, one could always weaken this formula later. In this subsection, we drop this assumption and also allow relevant formulas in the logic. In this case, contraction cannot always be done, as, once a relevant formula is contracted, it can no longer be weakened and thus must be used in some logical rule.

As before, we try to postpone as much as possible the contraction of relevant formulas. The contraction rule still permutes over all rules, except tensor rules and dereliction rules. For example, contraction permutes over $\parr$ rule:

$$\dfrac{\dfrac{\vdash \Gamma, A, B, ?^c P, ?^c P}{\vdash \Gamma, A \parr B, ?^c P, ?^c P}\,[\parr]}{\vdash \Gamma, A \parr B, ?^c P}\,[C?^c] \quad \rightsquigarrow \quad \dfrac{\dfrac{\vdash \Gamma, A, B, ?^c P, ?^c P}{\vdash \Gamma, A, B, ?^c P}\,[C?^c]}{\vdash \Gamma, A \parr B, ?^c P}\,[\parr]$$

However, there are some cases when contraction rules do permute over tensor and dereliction rules, namely when there are already at least two copies of the contracted formula, as illustrates the following transformations:

$$\dfrac{\dfrac{\vdash \Gamma, A, ?^c P, ?^c P, ?^c P \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B, ?^c P, ?^c P, ?^c P}\,[\otimes]}{\vdash \Gamma, \Delta, A \otimes B, ?^c P, ?^c P}\,[C?^c] \quad \rightsquigarrow \quad \dfrac{\dfrac{\vdash \Gamma, A, ?^c P, ?^c P, ?^c P}{\vdash \Gamma, A, ?^c P, ?^c P}\,[C?^c] \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B, ?^c P, ?^c P}\,[\otimes]$$

$$\dfrac{\dfrac{\vdash \Gamma, A, ?^c P, ?^c P \quad \vdash \Delta, B, ?^c P}{\vdash \Gamma, \Delta, A \otimes B, ?^c P, ?^c P, ?^c P}\,[\otimes]}{\vdash \Gamma, \Delta, A \otimes B, ?^c P, ?^c P}\,[C?^c] \quad \rightsquigarrow \quad \dfrac{\dfrac{\vdash \Gamma, A, ?^c P, ?^c P}{\vdash \Gamma, A, ?^c P}\,[C?^c] \quad \vdash \Delta, B, ?^c P}{\vdash \Gamma, \Delta, A \otimes B, ?^c P, ?^c P}\,[\otimes]$$

$$\dfrac{\dfrac{\vdash \Gamma, ?^c P, ?^c P, P}{\vdash \Gamma, ?^c P, ?^c P, ?^c P}\,[D?^c]}{\vdash \Gamma, ?^c P, ?^c P}\,[C?^c] \quad \rightsquigarrow \quad \dfrac{\dfrac{\vdash \Gamma, ?^c P, ?^c P, P}{\vdash \Gamma, ?^c P, P}\,[C?^c]}{\vdash \Gamma, ?^c P, ?^c P}\,[D?^c]$$

Hence, whenever there is more than one copy of a relevant formula, $?^c P$, in the context, one can treat these formulas as bounded formulas and split them in the tensor rule, postponing their contraction. Moreover, even when there is only one copy of a relevant formula, $?^c P$, in the context, one can permute its contraction over the tensor rule, namely when all copies goes to one of its branches, as illustrates the following transformation:

$$\dfrac{\dfrac{\vdash \Gamma, A, ?^c P, ?^c P \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B, ?^c P, ?^c P}\,[\otimes]}{\vdash \Gamma, \Delta, A \otimes B, ?^c P}\,[C?^c] \quad \rightsquigarrow \quad \dfrac{\dfrac{\vdash \Gamma, A, ?^c P, ?^c P}{\vdash \Gamma, A, ?^c P}\,[C?^c] \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B, ?^c P}\,[\otimes]$$

The only case when a contraction cannot permute over a tensor is when there is only one copy of the formula in the context and after contracting it, the copies are split by the tensor (illustrated by the derivation to the left); and the only case when a contraction cannot permute over a dereliction is when there is only one copy of the formula in the context and before applying the dereliction rule, this formula is contracted (illustrated by the derivation to the right):

$$\dfrac{\dfrac{\vdash \Gamma, A, ?^c P \quad \vdash \Delta, B, ?^c P}{\vdash \Gamma, \Delta, A \otimes B, ?^c P, ?^c P}\,[\otimes]}{\vdash \Gamma, \Delta, A \otimes B, ?^c P}\,[C?^c] \qquad \dfrac{\dfrac{\vdash \Gamma, ?^c P, P}{\vdash \Gamma, ?^c P, ?^c P}\,[D?^c]}{\vdash \Gamma, ?^c P}\,[C?^c]$$

Summarizing: for both the tensor and the dereliction rule, whenever there are two copies of a relevant formula, $?^c P$, in the context, we do not need to contract them; otherwise, if there is only one copy of

a relevant formula, then there are two options, either do not contract it, or contract it once and for the tensor rule, split the two copies among its branches.

We formalize this intuition as follows: assume that $l \in \mathcal{C} \setminus \mathcal{W}$, then $(\mathcal{K}_1 \otimes \mathcal{K}_2)[l]$ is a sub-multiset of the multiset $\mathcal{K}_1[l] \cup \mathcal{K}_2[l]$ such that all formulas in the multiset $\delta = \mathcal{K}_1[l] \cup \mathcal{K}_2[l] \setminus (\mathcal{K}_1 \otimes \mathcal{K}_2)[l]$ have multiplicity one in the multisets $\delta, \mathcal{K}_1[l], \mathcal{K}_2[l]$ and $(\mathcal{K}_1 \otimes \mathcal{K}_2)[l]$. Intuitively, δ represents the formulas that were created by contracting formulas. The conditions over the formulas in δ guarantee that only the formulas that appear once in $(\mathcal{K}_1 \otimes \mathcal{K}_2)[l]$ are contracted (given by the condition of having multiplicity one in $(\mathcal{K}_1 \otimes \mathcal{K}_2)[l]$). Moreover these formulas are contracted only once (given by the condition of having multiplicity one in δ) and the two copies are split in the functions $\mathcal{K}_1$ and $\mathcal{K}_2$ (given by the condition of multiplicity one in $\mathcal{K}_1[l]$ and $\mathcal{K}_2[l]$). We use the same tensor rule as in *SELLF*:

$$\frac{\vdash \mathcal{K}_1 : \Gamma \Downarrow A \quad \vdash \mathcal{K}_2 : \Delta \Downarrow B}{\vdash \mathcal{K}_1 \otimes \mathcal{K}_2 : \Gamma, \Delta \Downarrow A \otimes B} \ [\otimes, \text{ provided that } (\mathcal{K}_1 = \mathcal{K}_2) \mid_{\mathcal{C} \cap \mathcal{W}}]$$

Now, for the dereliction rule, if we do not want to contract a relevant formula, P, we use the following decide rule (that is already in *SELLF*):

$$\frac{\vdash \mathcal{K} : \Gamma \Downarrow P}{\vdash \mathcal{K} +_l P : \Gamma \Uparrow \cdot} \ [D_l, \text{ provided } l \notin \mathcal{C} \cap \mathcal{W}]$$

otherwise, if there is only one copy of P in the context, then we use alternatively the following decide rule that contracts P:

$$\frac{\vdash \mathcal{K} +_l P : \Gamma \Downarrow P}{\vdash \mathcal{K} +_l P : \Gamma \Uparrow \cdot} \ [D_l, \text{ provided } l \in \mathcal{C} \setminus \mathcal{W} \text{ and } \{P, P\} \nsubseteq \mathcal{K}[l]]$$

We call the system obtained by adding the rule above to $SELLF_\Sigma$ as $SELLF^+_\Sigma$. The following theorem is a direct consequence of the discussion above.

Theorem 5.14 *Let Σ be a subexponential signature. Then, $SELLF^+_\Sigma$ is sound and complete with respect to $SELL_\Sigma$.*

One could imagine even tighter conditions for when one does not need to contract relevant formulas. Consider, for example, the subexponential indexes $c \preceq d$ such that $c \in \mathcal{C} \setminus \mathcal{W}$ and $d \in \mathcal{C} \cap \mathcal{W}$, and that the formula $P \in \mathcal{K}[c] \cap \mathcal{K}[d]$. In this case, one does not need to contract $?^c P$ because one can always use the "safe" copies created by contracting the formula $?^d P$.

We now consider an alternative focused system for $SELL_\Sigma$, called $SELLF^C_\Sigma$, where the contractions of relevant formulas are not implicit in the logical rules, but occur in a third focusing phase, between the asynchronous and synchronous phases. The system $SELLF^C_\Sigma$ is obtained by adding the following rule to the system $SELLF_\Sigma$:

$$\frac{\vdash \mathcal{K} +_l P : \Gamma \Uparrow}{\vdash \mathcal{K} : \Gamma \Uparrow} \ [C?^c, \text{provided } P \in \mathcal{K}[l] \text{ and } l \in \mathcal{C} \setminus \mathcal{W}]$$

and for the tensor rule, we do not contract relevant formulas, but split then. This is formalized by defining $(\mathcal{K}_1 \otimes \mathcal{K}_2)[l]$, for $l \in \mathcal{C} \setminus \mathcal{W}$, as the multiset $\mathcal{K}_1[l] \cup \mathcal{K}_2[l]$. Hence, (from bottom-up) at the end of the asynchronous phase, one is allowed to either decide on a formula using the decide rules in $SELL_\Sigma$, or contract relevant formulas. This creates a new phase between the asynchronous and synchronous phases, similar to the cut phase discussed in Section 3.5 of Chapter 3.

This new system is complete because all synchronous rules permute over contractions, hence we can permute all contractions in a synchronous phase down to the beginning of this phase; and on the other hand, since contraction rules permute over all asynchronous rules, we permute all contractions appearing in the asynchronous phase up to the end of this phase. We show here some of the permutation cases involving synchronous rules and contractions:

$$\dfrac{\dfrac{\vdash \Gamma, A, ?^c P, ?^c P}{\vdash \Gamma, A, ?^c P}\,[C?^c] \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B, ?^c P}\,[\otimes] \quad \rightsquigarrow \quad \dfrac{\vdash \Gamma, A, ?^c P, ?^c P \quad \vdash \Delta, B}{\dfrac{\vdash \Gamma, \Delta, A \otimes B, ?^c P, ?^c P}{\vdash \Gamma, \Delta, A \otimes B, ?^c P}\,[C?^c]}\,[\otimes]$$

$$\dfrac{\dfrac{\vdash \Gamma, A, ?^c P, ?^c P}{\vdash \Gamma, A, ?^c P}\,[C?^c]}{\vdash \Gamma, A \oplus B, ?^c P}\,[\oplus_1] \quad \rightsquigarrow \quad \dfrac{\dfrac{\vdash \Gamma, A, ?^c P, ?^c P}{\vdash \Gamma, A \oplus B, ?^c P, ?^c P}\,[\oplus_1]}{\vdash \Gamma, A \oplus B, ?^c P}\,[C?^c]$$

$$\dfrac{\dfrac{\vdash \Gamma, A[t/x], ?^c P, ?^c P}{\vdash \Gamma, A[t/x], ?^c P}\,[C?^c]}{\vdash \Gamma, \exists x A, ?^c P}\,[\exists] \quad \rightsquigarrow \quad \dfrac{\dfrac{\vdash \Gamma, A[t/x], ?^c P, ?^c P}{\vdash \Gamma, \exists x A, ?^c P, ?^c P}\,[\exists]}{\vdash \Gamma, \exists x A, ?^c P}\,[C?^c]$$

$$\dfrac{\dfrac{\vdash \Gamma, A, ?^c P, ?^c P}{\vdash \Gamma, A, ?^c P}\,[C?^c]}{\vdash \Gamma, ?^d A, ?^c P}\,[D?^d] \quad \rightsquigarrow \quad \dfrac{\dfrac{\vdash \Gamma, A, ?^c P, ?^c P}{\vdash \Gamma, ?^d A, ?^c P, ?^c P}\,[D?^d]}{\vdash \Gamma, ?^d A, ?^c P}\,[C?^c]$$

Theorem 5.15 *Let Σ be a subexponential signature. Then, $SELLF^C_\Sigma$ is sound and complete with respect to $SELL_\Sigma$.*

It is worth noticing that we cannot permute all rules over contractions, and hence we cannot consider just one "contraction phase" appearing at the bottom of the tree. The problem is that the & rule does not permute over contractions, as illustrates the following derivation:

$$\dfrac{\dfrac{\vdash \Gamma, A, ?^c F, \ldots, ?^c F}{\vdash \Gamma, A, ?^c F}\,[m \times C?^c] \quad \dfrac{\vdash \Gamma, A, ?^c F, \ldots, ?^c F}{\vdash \Gamma, A, ?^c F}\,[n \times C?^c]}{\vdash \Gamma, A \,\&\, B, ?^c F}\,[\&]$$

Since the number of contractions of $?^c F$ is different in the two different branches (m times in the left branch and n times in the right branch), we cannot permute the & rule over all contractions.

5.3 Creating and modifying subexponentials

Until now we assumed that there is a global subexponential signature specifying the existing subexponentials and the relations among them. Now, we consider extending $SELL_\Sigma$ with new connectives, $\Cap$ and $\Cup$, that can change subexponential signatures of the logic or instantiate subexponentials. We call $SELL^{\Cap}$ the system that contains such connectives.

The sequents in $SELL^{\Cap}$ contain a subexponential signature declaration, as in the sequent $\Sigma \vdash \Gamma$, where Γ is a multiset of formulas and $\Sigma = \langle \mathcal{I}, \preceq, \mathcal{W}, \mathcal{C} \rangle$ a subexponential signature. The rules in $SELL^{\Cap}$ are depicted in Figure 5.3, where the following operations over subexponential signatures $\Sigma_1 = \langle \mathcal{I}_1, \preceq_1, \mathcal{W}_1, \mathcal{C}_1 \rangle$ and $\Sigma_2 = \langle \mathcal{I}_2, \preceq_2, \mathcal{W}_2, \mathcal{C}_2 \rangle$ are defined as follows:

- $\Sigma_1 \cup \Sigma_2$ is the tuple $\langle \mathcal{I}_1 \cup \mathcal{I}_2, \preceq_1 \cup \preceq_2, \mathcal{W}_1 \cup \mathcal{W}_2, \mathcal{C}_1 \cup \mathcal{C}_2 \rangle$;
- $\Sigma_1 \subseteq \Sigma_2$ iff $\mathcal{I}_1 \subseteq \mathcal{I}_2, \preceq_1 \subseteq \preceq_2, \mathcal{W}_1 \subseteq \mathcal{W}_2$, and $\mathcal{C}_1 \subseteq \mathcal{C}_2$.

Theorem 5.16 *The cut-elimination theorem holds for $SELL^{\Cap}$.*

Proof The proof is similar to the usual linear logic cut-elimination proof. Because of the new connectives $\Cap$ and $\Cup$, we have the following new key case:

$$\dfrac{\dfrac{\dfrac{\Xi_1}{\Sigma \cup \Sigma_l \vdash \Gamma, P}}{\Sigma \vdash \Gamma, \Cap\Sigma_l.P} \quad \dfrac{\dfrac{\Xi_2}{\Sigma \vdash \Delta, P^\perp\theta}}{\Sigma \vdash \Delta, \Cup\Sigma_l.P^\perp}}{\Sigma \vdash \Gamma, \Delta} \quad \rightsquigarrow \quad \dfrac{\dfrac{\Xi_1\theta}{\Sigma \vdash \Gamma, P\theta} \quad \dfrac{\Xi_2}{\Sigma \vdash \Delta, P^\perp\theta}}{\Sigma \vdash \Gamma, \Delta}$$

where $\theta = [s_1/l_1, \ldots, s_n/l_n]$.

It is easy to check that the derivation $\Xi_1\theta$ is indeed a proof. We show this by induction on the height of Ξ_1. The only interesting case is when the last rule is the promotion rule:

$$\dfrac{\Sigma \vdash \Gamma, C}{\Sigma \vdash \Gamma, !^c C}\ [!^c]$$

By the induction hypothesis, there is a proof of the sequent $(\Sigma \vdash \Gamma, C)\theta$. Assume without loss of generality that $?^{l_i}P \in \Gamma$, this would imply that $c \preceq l_i$ is valid in the signature Σ. Hence, it must be the case that $c \preceq s_i$ is valid in the signature $\Sigma\theta$, and therefore there is a proof of $(\Sigma \vdash \Gamma, !^c C)\theta$. The case when $c \in \{l_1, \ldots, l_n\}$ is similar. Then the proof proceeds as usual (see [Braüner 1996]). □

The focused system for $SELL^{\Cap}$, called $SELLF^{\Cap}$, is similar to the focused system $SELL^{+}_{\Sigma}$, just that the subexponential signatures are now declared in the sequents of the rules. For example, the tensor rule in $SELLF^{\Cap}$ is

$$\dfrac{\Sigma \vdash \mathcal{K}_1 : \Gamma \Downarrow A \quad \Sigma \vdash \mathcal{K}_2 : \Delta \Downarrow B}{\Sigma \vdash \mathcal{K}_1 \otimes \mathcal{K}_2 : \Gamma, \Delta \Downarrow A \otimes B}\ [\otimes,\ \text{provided that } (\mathcal{K}_1 = \mathcal{K}_2)\ |_{\mathcal{C} \cap \mathcal{W}}]$$

Moreover, we add the following two rules for the new connectives $\Cap$ and $\Cup$:

$$\dfrac{\Sigma \cup \Sigma_l \vdash \mathcal{K} : \Gamma \Uparrow L, C}{\Sigma \vdash \mathcal{K} : \Gamma \Uparrow L, \Cap\Sigma_l.C}\ [\Cap] \qquad \dfrac{\Sigma \vdash \mathcal{K} : \Gamma \Downarrow C[s_1/l_1, \ldots, s_n/l_n]}{\Sigma \vdash \mathcal{K} : \Gamma \Downarrow \Cup\Sigma_l.C}\ [\Cup]$$

with the proviso that $\Sigma \cup \Sigma_l$ is a consistent subexponential signature for the $\Cap$ rule, and the proviso that $\Sigma_l[s_1/l_1, \ldots, s_n/l_n] \subseteq \Sigma$ for the $\Cup$ rule.

We use once more foculisation graphs to prove that $SELLF^{\Cap}$ is complete. We classify the connective $\Cap$ as asynchronous and $\Cup$ as synchronous. The remaining definitions (for the foculisation graph, $\prec_f$) are used as before. We continue showing the permutation lemmas.

Lemma 5.17 *Let α be an inference rule and β an asynchronous rule. Then α/β.*

Proof We show only some of the cases involving the new connectives. Notice that augmenting the subexponential signature does not affect provability, one just has to take care to rename the "bounded" indexes when needed.

- $(\otimes/\Cap)$:

$$\dfrac{\dfrac{\Sigma \cup \Sigma_l \vdash \mathcal{K}_1 : \Gamma, A, P}{\Sigma \vdash \mathcal{K}_1 : \Gamma, A, \Cap\Sigma_l.P}\ [\Cap] \quad \Sigma \vdash \mathcal{K}_2 : \Delta, B}{\Sigma \vdash \mathcal{K}_1 \otimes \mathcal{K}_2 : \Gamma, \Delta, A \otimes B, \Cap\Sigma_l.P}\ [\otimes] \quad \rightsquigarrow \quad \dfrac{\dfrac{\Sigma \cup \Sigma_l \vdash \mathcal{K}_1 : \Gamma, A, P \quad \Sigma \cup \Sigma_l \vdash \mathcal{K}_2 : \Delta, B}{\Sigma \cup \Sigma_l \vdash \mathcal{K}_1 \otimes \mathcal{K}_2 : \Gamma, \Delta, A \otimes B, P}\ [\otimes]}{\Sigma \vdash \mathcal{K}_1 \otimes \mathcal{K}_2 : \Gamma, \Delta, A \otimes B, \Cap\Sigma_l.P}\ [\Cap]$$

- $(\&/\Cap)$:

$$\dfrac{\dfrac{\Sigma \cup \Sigma_l \vdash \mathcal{K} : \Gamma, A, P}{\Sigma \vdash \mathcal{K} : \Gamma, A, \Cap\Sigma_l.P}\ [\Cap] \quad \dfrac{\Sigma \cup \Sigma_l \vdash \mathcal{K} : \Gamma, B, P}{\Sigma \vdash \mathcal{K} : \Gamma, B, \Cap\Sigma_l.P}\ [\Cap]}{\Sigma \vdash \mathcal{K} : \Gamma, A \,\&\, B, \Cap\Sigma_l.P}\ [\&] \quad \rightsquigarrow$$

$$\dfrac{\dfrac{\Sigma \cup \Sigma_l \vdash \mathcal{K} : \Gamma, A, P \quad \Sigma \cup \Sigma_l \vdash \mathcal{K} : \Gamma, B}{\Sigma \cup \Sigma_l \vdash \mathcal{K} : \Gamma, A \,\&\, B, P}\ [\&]}{\Sigma \vdash \mathcal{K} : \Gamma, A \,\&\, B, \Cap\Sigma_l.P}\ [\Cap]$$

IDENTITY RULES

$$\frac{}{\Sigma \vdash P, P^{\perp}}\ [I] \qquad \frac{\Sigma \vdash \Gamma, P^{\perp} \quad \Sigma \vdash \Delta, P}{\Sigma \vdash \Gamma, \Delta}\ [Cut]$$

LOGICAL RULES

$$\frac{}{\Sigma \vdash 1}\ [1] \qquad \frac{}{\Sigma \vdash \Gamma, \top}\ [\top] \qquad \frac{\Sigma \vdash \Gamma}{\Sigma \vdash \Gamma, \perp}\ [\perp]$$

$$\frac{\Sigma \vdash \Gamma, P_i}{\Sigma \vdash \Gamma, P_1 \oplus P_2}\ [\oplus_i] \qquad \frac{\Sigma \vdash \Gamma, P \quad \Sigma \vdash \Gamma, Q}{\Sigma \vdash \Gamma, P \,\&\, Q}\ [\&]$$

$$\frac{\Sigma \vdash \Gamma, P, Q}{\Sigma \vdash \Gamma, P \parr Q}\ [\parr] \qquad \frac{\vdash \Sigma; \Gamma, P \quad \Sigma \vdash \Delta, Q}{\Sigma \vdash \Gamma, \Delta, P \otimes Q}\ [\otimes]$$

$$\frac{\Sigma \vdash \Gamma, P[t/x]}{\Sigma \vdash \Gamma, \exists x.P}\ [\exists] \qquad \frac{\Sigma \vdash \Gamma, P[c/x]}{\Sigma \vdash \Gamma, \forall x.P}\ [\forall]$$

$$\frac{\Sigma \vdash \Gamma, P}{\vdash \Sigma; \Gamma, ?^{c}P}\ [D?^{c}, \text{ provided } c \in \mathcal{I}]$$

$$\frac{\Sigma \vdash ?^{x_1}C_1, \ldots, ?^{x_n}C_n, C}{\Sigma \vdash ?^{x_1}C_1, \ldots, ?^{x_n}C_n, !^{c}C}\ [!^{c}, \text{ provided } x_i, c \in \mathcal{I} \text{ and } c \preceq x_i, \text{ for all } i = 1, \ldots, n]$$

$$\frac{\Sigma \cup \Sigma_l \vdash \Gamma, C}{\Sigma \vdash \Gamma, \Subset \Sigma_l.C}\ [\Subset, \text{ if } \Sigma \cup \Sigma_l \text{ is consistent}]$$

$$\frac{\Sigma \vdash \Gamma, C[s_1/l_1, \ldots, s_n/l_n]}{\Sigma \vdash \Gamma, \Supset \Sigma_l.C}\ [\Supset, \text{ if } \Sigma_l[s_1/l_1, \ldots, s_n/l_n] \subseteq \Sigma]$$

STRUCTURAL RULES

$$\frac{\Sigma \vdash \Gamma, ?^{c}P, ?^{c}P}{\Sigma \vdash \Gamma, ?^{c}P}\ [C?^{c}, \text{provided } c \in \mathcal{C}] \qquad \frac{\Sigma \vdash \Gamma}{\Sigma \vdash \Gamma, ?^{c}P}\ [W?^{c}, \text{provided } c \in \mathcal{W}]$$

Figure 5.3: The inference rules for $SELL^{\Subset}$. Here $\Sigma = \langle \mathcal{I}, \preceq, \mathcal{W}, \mathcal{C} \rangle$ and $\Sigma_l = \langle \mathcal{I}_l, \preceq_l, \mathcal{W}_l, \mathcal{C}_l \rangle$ are subexponential signatures, such that $\mathcal{I}_l = \{l_1, \ldots, l_n\}$ is a set of new subexponentials, the preorder $\preceq_l \subseteq \hat{\mathcal{I}} \times \hat{\mathcal{I}}$, the sets of indexes $\mathcal{W}_l, \mathcal{C}_l \subseteq \hat{\mathcal{I}}$, and $\hat{\mathcal{I}} = \mathcal{I}_l \cup \mathcal{I}$ is set of subexponential indexes.

- $(\Cup/\Cap)$:

$$\dfrac{\dfrac{\Sigma \cup \Sigma_l \vdash \mathcal{K} : \Gamma, P\theta, Q}{\Sigma \vdash \mathcal{K} : \Gamma, P\theta, \Cap\Sigma_l.Q}\,[\Cap]}{\Sigma \vdash \mathcal{K} : \Gamma, \Cup\Sigma'_l.P, \Cap\Sigma_l.Q}\,[\Cup] \quad \rightsquigarrow \quad \dfrac{\dfrac{\Sigma \cup \Sigma_l \vdash \mathcal{K} : \Gamma, P\theta, Q}{\Sigma \cup \Sigma_l \vdash \mathcal{K} : \Gamma, \Cup\Sigma'_l.P\theta, Q}\,[\Cup]}{\Sigma \vdash \mathcal{K} : \Gamma, \Cup\Sigma'_l.P, \Cap\Sigma_l.Q}\,[\Cap]$$

- $(\Cap/\Cap)$:

$$\dfrac{\dfrac{\Sigma \cup \Sigma'_l \cup \Sigma_l \vdash \mathcal{K} : \Gamma, P, Q}{\Sigma \cup \Sigma'_l \vdash \mathcal{K} : \Gamma, P, \Cap\Sigma_l.Q}\,[\Cap]}{\Sigma \vdash \mathcal{K} : \Gamma, \Cap\Sigma'_l.P, \Cap\Sigma_l.Q}\,[\Cap] \quad \rightsquigarrow \quad \dfrac{\dfrac{\Sigma \cup \Sigma_l \cup \Sigma'_l \vdash \mathcal{K} : \Gamma, P, Q}{\Sigma \cup \Sigma_l \vdash \mathcal{K} : \Gamma, \Cap\Sigma'_l.P\theta, Q}\,[\Cap]}{\Sigma \vdash \mathcal{K} : \Gamma, \Cap\Sigma'_l.P, \Cap\Sigma_l.Q}\,[\Cap]$$

where $P\theta$ is a substitution instantiating some subexponential indexes in P. □

Lemma 5.18 *Let α and β be synchronous rules. Then α/β.*

Proof We only show some of the cases involving the new connective $\Cup$.

- $(\otimes/\Cup)$:

$$\dfrac{\dfrac{\Sigma \vdash \mathcal{K}_1 : \Gamma, A, P\theta}{\Sigma \vdash \mathcal{K}_1 : \Gamma, A, \Cup\Sigma_l.P}\,[\Cup] \quad \Sigma \vdash \mathcal{K}_2 : \Delta, B}{\Sigma \vdash \mathcal{K}_1 \otimes \mathcal{K}_2 : \Gamma, \Delta, A \otimes B, \Cup\Sigma_l.P}\,[\otimes] \quad \rightsquigarrow \quad \dfrac{\Sigma \vdash \mathcal{K}_1 : \Gamma, A, P\theta \quad \Sigma \vdash \mathcal{K}_2 : \Delta, B}{\Sigma \vdash \mathcal{K}_1 \otimes \mathcal{K}_2 : \Gamma, \Delta, A \otimes B, P\theta}\,[\otimes]}{\Sigma \vdash \mathcal{K}_1 \otimes \mathcal{K}_2 : \Gamma, \Delta, A \otimes B, \Cup\Sigma_l.P}\,[\Cup]$$

- $(\Cup/\oplus_1)$:

$$\dfrac{\dfrac{\Sigma \vdash \mathcal{K} : \Gamma, P\theta, A}{\Sigma \vdash \mathcal{K} : \Gamma, P\theta, A \oplus B}\,[\oplus_1]}{\Sigma \vdash \mathcal{K} : \Gamma, \Cup\Sigma_l.P, A \oplus B}\,[\Cup] \quad \rightsquigarrow \quad \dfrac{\dfrac{\Sigma \vdash \mathcal{K} : \Gamma, P\theta, A}{\Sigma \vdash \mathcal{K} : \Gamma, \Cup\Sigma_l.P, A}\,[\Cup]}{\Sigma \vdash \mathcal{K} : \Gamma, \Cup\Sigma_l.P, A \oplus B}\,[\oplus_1]$$

- $(\Cup/\Cup)$:

$$\dfrac{\dfrac{\Sigma \vdash \mathcal{K} : \Gamma, P\theta, Q\theta'}{\Sigma \vdash \mathcal{K} : \Gamma, P\theta, \Cup\Sigma'_l.Q}\,[\Cup]}{\Sigma \vdash \mathcal{K} : \Gamma, \Cup\Sigma_l.P, \Cup\Sigma'_l.Q}\,[\Cup] \quad \rightsquigarrow \quad \dfrac{\dfrac{\Sigma \vdash \mathcal{K} : \Gamma, P\theta, Q\theta'}{\Sigma \vdash \mathcal{K} : \Gamma, \Cup\Sigma_l.P, Q\theta'}\,[\Cup]}{\Sigma \vdash \mathcal{K} : \Gamma, \Cup\Sigma_l.P, \Cup\Sigma'_l.Q}\,[\Cup]$$

□

It is easy to check (with the same reasoning as in Lemma 2.16) that the new connectives do not affect the acyclicity of the relation $\prec_f$.

Lemma 5.19 *The relation $\prec_f$ is acyclic.*

The following soundness and completeness theorem follows, as in Theorem 2.8, by using the permutation lemmas above and the fact that $\prec_f$ is acyclic.

Theorem 5.20 *SELLF$^{\Cap}$ is sound and complete with respect to SELL$^{\Cap}$.*

5.4 Related Works

Schellinx shows in [Schellinx 1993] that one can capture the dynamics of cut-elimination of the modal logic S4 by using subexponentials. We need two colors of subexponentials, say 0 and 1, where the subexponential 0 is bounded and is weaker than the unbounded subexponential 1. Intuitively, the subexponential 0 is used to denote the modals $\Diamond$ and $\Box$ in S4, and the subexponential 1 is used to denote the usual linear logic exponentials. Then, Schellinx proposes a translation of modal logic

formulas, which uses these subexponentials, for which the dynamics of cut-elimination in S4 are captured in linear logic.

Elsewhere [Hernest 2008], Hernest and Oliva used two subexponentials, k and g, to mix two different functional interpretations of linear logic formulas, namely the unbounded subexponential k for Kreisel's modified realizability [Kreisel 1959] and the unbounded subexponential g for Gödel's Dialectica interpretation [Gödel 1958], where $k \preceq g$. In this hybrid interpretation, one can combine features of these different functional interpretations, such as the extensionality schema rule $x \stackrel{\rho}{=} y \to fx \stackrel{\tau}{=} fy$, by using Kreisel's modified interpretation, or Markov's principle, $\neg\forall x.A(x) \to \exists x.\neg A(x)$, by using Gödel's dialectica interpretation.

Closer to the contents of this chapter, Hodas proposed in his PhD thesis [Hodas 1994b, Chapter 8] an intuitionistic linear logic system, called Omnibus Logic, containing only one subexponential index for affine, relevant, and unbounded formulas[2]. These indexes were implicit in the logic by the presence of different contexts, as in our polyadic system, in the left-hand-side of sequents: $\Theta_u; \Theta_r; \Theta_a; \Gamma \longrightarrow C$. Here, Θ_u is a multiset of unbounded formulas; Θ_r is a multiset of relevant formulas; Θ_a is a multiset of affine formulas; and Γ a multiset of bounded formulas. Hodas also used different implications for each type of formula: the usual intuitionistic implication $\supset$, the affine implication $\stackrel{a}{\to}$, the relevant implication $\stackrel{r}{\to}$, and the linear implication $\multimap$. The translation of the non-linear implications to linear logic is done as in Girard's translation (see end of Section 2.4), but by using the corresponding subexponential bang for each type of implication. Hodas assumed no relation between the subexponential indexes, so for example, the rule introducing an affine implication on the left was of the form:

$$\frac{\Theta_u; \cdot; \Theta^1_a; \cdot \longrightarrow P \quad \Theta_u; \Theta_r; \Theta^2_a; \Gamma, Q \longrightarrow C}{\Theta_u; \Theta_r; \Theta^1_a, \Theta^2_a; \Gamma, P \stackrel{a}{\to} Q \longrightarrow C}\ [\stackrel{a}{\to}]$$

Notice that, as in our polyadic system, the formulas in the unbounded context are contracted, while the formulas in the affine context are split among the branches.

The treatment of unbounded, affine and bounded formulas in Hodas' system is close to the treatment in our polyadic system. However, when it comes to the relevant formulas, Hodas uses the following unsound rule, with respect to *SELL*, for dereliction of relevant formulas:

$$\frac{\Theta_u, F; \Theta_r, \Theta_a; \Gamma, F \longrightarrow C}{\Theta_u; \Theta_r, F, \Theta_a; \Gamma \longrightarrow C}\ [abs_R]$$

The problem occurs when we use this rule in combination with the affine bang rule, implicit in the implication left introduction rule above. The contraction of the relevant formula F, which, in Hodas's system, is performed in the unbounded context, is done in *SELL* in the relevant context. Moreover, since the subexponential indexes are unrelated, when introducing an affine bang, one must be sure that the relevant context is empty, which is not necessarily true with the use of the abs_R rule above,

[2] The fragment of this system, without the relevant subexponential, was also used by Chaudhuri in his PhD thesis [Chaudhuri 2006, Chapter 3].

as illustrates the following "proof" in Hodas' system:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{F;\cdot;\cdot;C \longrightarrow C}\,[I] \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{F;\cdot;\cdot;D \longrightarrow D}\,[I]}{F;\cdot;P,P \otimes P \xrightarrow{a} D;C \xrightarrow{a} D \longrightarrow D}\,[abs_A]}{F;\cdot;P,C \xrightarrow{a} D,P \otimes P \xrightarrow{a} D;\cdot \longrightarrow D}\,[\xrightarrow{a}_r]}{F;\cdot;C \xrightarrow{a} D,P \otimes P \xrightarrow{a} D;\cdot \longrightarrow P \xrightarrow{a} D}\,[\multimap_l]}{F;\cdot;C \xrightarrow{a} D,P \otimes P \xrightarrow{a} D;(P \xrightarrow{a} D) \multimap C \longrightarrow C}\,[abs_R]}{\cdot;(P \xrightarrow{a} D) \multimap C;C \xrightarrow{a} D,P \otimes P \xrightarrow{a} D;\cdot \longrightarrow C} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{F;\cdot;P,P;P \otimes P \xrightarrow{a} D \longrightarrow D}\,[abs_A]}{F;\cdot;P,P,P \otimes P \xrightarrow{a} D;\cdot \longrightarrow D}}{F;\cdot;P,P \otimes P \xrightarrow{a} D;(P \xrightarrow{a} D) \multimap C \longrightarrow C}\,[abs_I]}{F;\cdot;P,P \otimes P \xrightarrow{a} D;\cdot \longrightarrow C}\,[\star \xrightarrow{a}_l]$$

where $F = (P \xrightarrow{a} C) \multimap C$. Notice that the introduction of the left affine implication in $\star$ is not sound, because in fact the contraction of the formula F, done in the rule abs_I, would have to occur before $\star$ (seeing bottom-up) and, therefore, the relevant context would not be empty.

Finally, Chaudhuri in [Chaudhuri 2009] showed that the propositional multiplicative fragment of *SELL* with three subexponentials (two bounded and one unbounded) can encode the transitions of the two-register Minsky machine [Minsky 1961], which is Turing complete, showing, hence, that this fragment of *SELL* is undecidable.

5.5 Conclusions and future works

In this chapter, we discussed the linear logic system, *SELL* proposed by Danos *et al.* [Danos 1993], containing *subexponentials*, and then we proposed focusing systems for *SELL*. First, we considered the case where relevant formulas are not permitted, proposing a straightforward extension to Andreoli's LLF system, called *SELLF*. Later, we investigated the general case and we proposed two complete focused systems for *SELL*: the first one where the structural rules are implicit in the logical rules and the second one where contractions of relevant formulas appear in a third focusing phase between asynchronous and synchronous phases. Finally, we proposed the system $SELL^{\Cap}$, an extension of linear logic, where the subexponential signature is not assumed to be global but local in the sequents and that contains two new connectives, $\Cup$ and $\Cap$, that can instantiate or extend such signatures. We showed that the cut-elimination theorem holds in $SELL^{\Cap}$ and later we proposed a complete focused version of this system called $SELLF^{\Cap}$.

We now point out some possible directions for future work.

- Liang & Miller in [Liang 2009] proposed the focused system *LKU* where linear, classical, and intuitionistic logics coexist in a unified system. Similarly to *LU* [Girard 1991], one can access these logics by assigning polarities to the connectives and atoms of formulas: for example, linear logic is captured if all connectives are assigned +/- 1 polarity, while classical logic is captured if all connectives are assigned +/- 2 polarity. These two levels of polarities are quite natural, as Liang & Miller used Andreoli's LLF focused system for linear logic to develop *LKU*. However, as already forecasted by Girard [Girard 1991], there can be infinitely many polarities. A direction for future work could be to use, say *SELLF*, to construct a unified focused proof system where such a range of polarity assignments exists. There are several design choices to consider:

  One could consider the design used by Liang & Miller [Liang 2009], where the polarity of the formulas would specify in which "context" positive formulas are moved to, while negative formulas are decomposed in the asynchronous phase. They also consider specialized systems

that links the polarity assignment for literals to the context where they belong: for example, a sequent that focuses on a $+2$ polarity literal, A, must be the conclusion of an instance of the initial rule where the matching literal, $A^{\perp}$, appears in the classical context. These type of systems have great appeal, as often in the computation-as-proof-search paradigm, one needs the following assumption for the specification of algorithms using focused proof systems (see Chapter 7 or [Simmons 2009]): the set of predicate names are divided into disjoint subsets where each subset contains the name of predicates allowed in a particular context of polyadic sequents. This assumption allows one to use the focusing initial rule and guarantee that the literal consumed belongs to a particular context. A possible way to express such assumption in the logic could be to link polarities to locations of literals.

On the other hand, as the equivalence $!^{k}A \multimap !^{i}A$ is provable in *SELL*, where the subexponentials $i, k \in \mathcal{I}$ and $i \preceq k$, one could choose the design closer to *LU* where contexts can contain any type of formulas and they can move from a context with a higher index to a context with a lower index.

- Although proof theory forecasts focused proof systems with relevant formulas, we have failed to find good computer science related examples that require such formulas. Such examples could help us decide which is the better design for a focused proof system of *SELL* containing relevant formulas. Here we have identified two such systems, one where the contraction of relevant formulas are implicit in the logical rules, and another where these contractions are isolated in a third focusing phase between asynchronous and synchronous phases.

- Here we have considered the case where subexponentials have only two degrees of structural rules, namely one for contraction and another for weakening. A direction for future work could be to extend *SELL* to contain subexponentials that do not allow exchange rules, and hence encode logics such as ordered logics [Polakow 2001].

Chapter 6

Focusing in linear meta logic with subexponentials

In this chapter we explore the increase of expressiveness obtained by using linear logic with subexponentials as a meta-logic. We show that a wide range of proof systems can be encoded in *SELLF* by using subexponentials to incorporate the structural restrictions of several proof systems for minimal, intuitionistic and classical logic, in a similar way as we did in Chapter 4, achieving always the strongest level of adequacy: *full completeness of derivations*. We encode Gentzen's *G1* system, Maehara's multiconclusion system for intuitionistic logic and several focused proof systems, such as Herbelin's LJQ^* system and Liang & Miller's *LJF* for intuitionistic logic and *LKF* for classical logic.

6.1 Introduction

When designing a proof system, the structural restrictions imposed to its sequents, usually through structural rules, play a role as important as the logical rules. Already in the first sequent calculus systems designed by Gentzen [Gentzen 1969], the system for intuitionistic logic differed from the system for classical logic by restricting in former system the right-hand-side of sequents to at most one formula. Since then, several other proof systems have been proposed, which differ more on these structural restrictions than on their logical rules. For example, Maehara [Maehara 1954] proposed the sequent calculus system for intuitionistic logic *mLJ*, where the right-hand-side of sequents are allowed to contain more than one formula, while the premise of the implication and of the universal quantifier introduction right rules is restricted to contain only one formula on its right-hand-side.

In Chapter 4, we used the focused linear logic system LLF to encode proof systems. There the structural restrictions of proof systems were captured by the use of exponentials. When trying to encode other proof system, however, we quickly stumble on the limitations of these exponentials. For example, it does not seem possible to encode sequent calculus systems, called *G1* in [Troelstra 1996], that restrict the contexts on the left and on the right of the turnstyle to be multisets and contain explicit structural rules for contraction and weakening. This is because linear logic without subexponentials provides only one linear context, while we would need two linear contexts to encode such systems: one to store the object-logic formulas on the right-hand-side of the turnstyle and another to store the object-logic formulas on the left-hand-side. For similar reasons we are not able to encode focused systems that contain not only the contexts for the left and the right-hand-side of the sequent, but also a context, called *stoup*, for the formula that is focused on.

In this chapter, we use the system *SELLF*, presented in Chapter 5, to encode sequent calculus systems. We show that, additionally to the systems encoded in Chapter 4, we can encode a wide range of more complicated proof systems with the adequacy of full derivations (see Subsection 4.2.2). After some preliminary considerations, we start by proposing theories that encode the sequent calculus systems *G1* for minimal, intuitionistic and classical logics [Troelstra 1996], where the structural rules for contraction and weakening are explicit. In Section 6.4, we describe a theory that encodes the multi-conclusion system for intuitionistic logic, *mLJ*, proposed by Maehara [Maehara 1954], which, differently from Gentzen's intuitionistic system, allows the right-hand-side of sequents to contain more than one formula. Next in Section 6.5, we propose a theory for LJQ^*, a focused system based on

mLJ and designed by Herbelin [Herbelin 1995, page 78] also appearing in [Dyckhoff 2006]. Then in Sections 6.6 and 6.7, we encode the focused sequent calculus systems *LKF* and *LJF*, for classical and intuitionistic logics, proposed by Liang & Miller in [Liang 2008, Liang 2007]. Finally, in the last two sections we discuss how to encode other focused proof systems and point out directions for future work.

6.2 Preliminaries

As in Chapter 4, we propose theories that use, instead of exponentials, subexponentials in such a way that the set of (open) derivations in *SELLF* obtained from these theories are in one to one correspondence with the set of (open) derivations of the encoded proof systems. These linear logic theories will, as in Chapter 4, contain two meta-level atoms, $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$, denoting the two senses of formulas in sequent calculus, that is, formulas on the left and right-hand-side of sequents (see Subsection 4.2.1).

To store these theories in sequents, we assume throughout this chapter the existence of a maximal subexponential, denoted by ∞, that is greater than all other subexponentials used. Moreover, to better illustrate the dynamics of formulas in *SELLF* proofs, we explicitly show the formulas in the images of the functions in their polyadic sequents. For example, assume that the set of subexponentials is $\{x1, \ldots, xn, \infty\}$, then the following sequent:

$$\vdash \Theta \stackrel{\cdot}{x1} \Theta_1 \stackrel{\cdot}{x2} \cdots \stackrel{\cdot}{xn} \Theta_n : \Gamma \Uparrow L$$

denotes the *SELLF* sequent $\vdash \mathcal{K} : \Gamma \Uparrow L$, where $\mathcal{K}[\infty] = \Theta$ and $\mathcal{K}[xi] = \Theta_i$. Moreover, we use the judgment $\vdash_{sellf}$ to denote provability in *SELLF*.

In Sections 6.6 and 6.7, where we encode the focused systems *LKF* and *LJF*, we will need a proof theoretic notion for fixed points, similar to the one introduced in Section 3.8, called definitions. They will be used to specify the polarity of object-logic formulas. A *definition* is a finite set of *clauses* which are written as $\forall\bar{x}[p\,\bar{x} \stackrel{\Delta}{=} B]$: here p is a predicate and every free variable of B (the *body* of the clause) is contained in the list $\bar{x}$. The symbol $\stackrel{\Delta}{=}$ is not a logical connective but is used to indicate a definitional clause. We consider that every defined predicate occurs at the head of exactly one clause. The following two "unfolding" rules are added to *SELLF*.

$$\frac{\vdash \mathcal{K} : \Gamma \Downarrow B\theta}{\vdash \mathcal{K} : \Gamma \Downarrow p\,\bar{t}}\ [def\Downarrow] \qquad \frac{\vdash \mathcal{K} : \Gamma \Uparrow L, B\theta}{\vdash \mathcal{K} : \Gamma \Uparrow L, p\,\bar{t}}\ [def\Uparrow]$$

The proviso for both of these rules is: $\forall\bar{x}[p\,\bar{x} \stackrel{\Delta}{=} B]$ is a definition clause and θ is the substitution that maps the variables $\bar{x}$ to the terms $\bar{t}$, respectively. Thus, in either phase of focusing, if a defined atom is encountered, it is simply replaced by its definition and the proof search phase does not change.

Notice that one could use, instead of *definitions*, the μ operator and the right introduction rule for equality shown in Section 3.8. However, we prefer the simpler presentation, as, in fact, definitions are not in the spot light of this chapter, but they are just used as a simple proof theoretic tool to illustrate the encodings of proof systems via subexponentials.

6.3 G1{mic}

We now encode the sequent calculus systems *G1m*, for minimal logic, *G1i*, for intuitionistic logic, and *G1c*, for classical logic [Troelstra 1996], which contain explicit structural rules for contraction and weakening and *multiplicative* logical rules[1]. The system *G1c* is depicted in Figure 6.1, and we

[1] These systems are called *G1* with *context-independent* rules in [Troelstra 1996, Remark 3.1.5].

$$\frac{\Gamma_1 \vdash \Delta_1, A \quad \Gamma_2, B \vdash \Delta_2}{\Gamma_1, \Gamma_2, A \Rightarrow B \vdash \Delta_1, \Delta_2}\ [\Rightarrow L] \qquad \frac{\Gamma, A \vdash \Delta, B}{\Gamma \vdash \Delta, A \Rightarrow B}\ [\Rightarrow R]$$

$$\frac{\Gamma, A_i \vdash \Delta}{\Gamma, A_1 \wedge A_2 \vdash \Delta}\ [\wedge_i L] \qquad \frac{\Gamma_1 \vdash \Delta_1, A \quad \Gamma_2 \vdash \Delta_2, B}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A \wedge B}\ [\wedge R]$$

$$\frac{\Gamma_1, A \vdash \Delta_1 \quad \Gamma_2, B \vdash \Delta_2}{\Gamma_1, \Gamma_2, A \vee B \vdash \Delta_1, \Delta_2}\ [\vee L] \qquad \frac{\Gamma \vdash \Delta, A_i}{\Gamma \vdash \Delta, A_1 \vee A_2}\ [\vee_i R]$$

$$\frac{\Gamma, A\{t/x\} \vdash \Delta}{\Gamma, \forall x\, A \vdash \Delta}\ [\forall L] \qquad \frac{\Gamma \vdash \Delta, A\{c/x\}}{\Gamma \vdash \Delta, \forall x\, A}\ [\forall R]$$

$$\frac{\Gamma, A\{c/x\} \vdash \Delta}{\Gamma, \exists x\, A \vdash \Delta}\ [\exists L] \qquad \frac{\Gamma \vdash \Delta, A\{t/x\}}{\Gamma \vdash \Delta, \exists x\, A}\ [\exists R]$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\ [W_L] \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash A, \Delta}\ [W_R]$$

$$\frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\ [C_L] \qquad \frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta}\ [C_R]$$

$$\frac{}{A \vdash A}\ [\mathrm{I}] \qquad \frac{\Gamma_1 \vdash \Delta_1, A \quad \Gamma_2, A \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\ [\mathrm{Cut}] \qquad \frac{}{\bot \vdash .}\ [\bot L]$$

Figure 6.1: The sequent calculus system *G1c* for classical logic. Here, $\Gamma_1, \Gamma_2, \Delta_1$ and Δ_2 are multisets of formulas; in the rules $\exists L$ and $\forall R$, the eigenvariable c does not appear free in Γ nor Δ; and $i \in \{1, 2\}$.

denote provability in this system by the judgment $\vdash_{g1c}$. From *G1c* we derive *G1i* by restricting the right-hand-side of sequents to at most one formula and replacing the introduction rules for implication by the following rules:

$$\frac{\Gamma_1 \vdash A \qquad \Gamma_2, B \vdash C}{\Gamma_1, \Gamma_2, A \supset B \vdash C}\ [\supset_l] \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B}\ [\supset_r]$$

While the system *G1m* is the system *G1i* without the rule $\bot_L$. We use the judgments $\vdash_{g1m}$ and $\vdash_{g1i}$ to denote provability in *G1m* and *G1i*, respectively.

To encode these systems, we use the following theories $\mathcal{L}_{g1m}$, $\mathcal{L}_{g1i}$ and $\mathcal{L}_{g1c}$ obtained from the theory $\mathcal{L}_{g1}$ in Figure 6.2, the formulas $(\supset_L), (\supset_R)$ and Id'_2 in Figure 6.3, and the identity and structural rules in Figure 6.4:

$$\begin{aligned}
&\mathcal{L}_{g1m} = (\mathcal{L}_{g1} \cup \{Id_1, Id'_2, W_L, C_L, (\supset_L), (\supset_R)\}) \setminus \{(\Rightarrow_L), (\Rightarrow_R), (\bot_L)\}\\
&\mathcal{L}_{g1i} = \mathcal{L}_{g1m} \cup \{W_R, (\bot_L)\}\\
&\mathcal{L}_{g1c} = \mathcal{L}_{g1} \cup \{Id_1, Id_2, C_L, C_R, W_L, W_R\}.
\end{aligned}$$

These theories use two unrelated and bounded subexponentials, l and r, where $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ meta-level atoms are going to be placed. The theory $\mathcal{L}_{g1m}$, for *G1m*, does not contain structural rules for $\lceil\cdot\rceil$ meta-level atoms and does not contain the formula $(\bot_L)$, while the theory $\mathcal{L}_{g1i}$, for *G1i*, contains the formula W_R that allows to consume $\lceil\cdot\rceil$ formulas and the formula $(\bot_L)$. Finally, the theory $\mathcal{L}_{g1c}$, for *G1c*, contains all structural rules. Moreover, the formulas $(\supset_L)$ and Id'_2 in the theories $\mathcal{L}_{g1m}$ and $\mathcal{L}_{g1i}$ have some occurrences of $!^l$, which are necessary to enforce that the right-hand-side of sequents contain at most one formula. For each one of these theories, we obtain the strongest level of adequacy by assigning negative polarity to all meta-level atoms.

$$
\begin{array}{llll}
(\Rightarrow_L) & \lfloor A \Rightarrow B \rfloor^\perp \otimes (?^r \lceil A \rceil \otimes ?^l \lfloor B \rfloor) & (\Rightarrow_R) & \lceil A \Rightarrow B \rceil^\perp \otimes (?^l \lfloor A \rfloor \parr ?^r \lceil B \rceil) \\
(\wedge_L) & \lfloor A \wedge B \rfloor^\perp \otimes (?^l \lfloor A \rfloor \oplus ?^l \lfloor B \rfloor) & (\wedge_R) & \lceil A \wedge B \rceil^\perp \otimes (?^r \lceil A \rceil \otimes ?^r \lceil B \rceil) \\
(\vee_L) & \lfloor A \vee B \rfloor^\perp \otimes (?^l \lfloor A \rfloor \otimes ?^l \lfloor B \rfloor) & (\vee_R) & \lceil A \vee B \rceil^\perp \otimes (?^r \lceil A \rceil \oplus ?^r \lceil B \rceil) \\
(\forall_L) & \lfloor \forall B \rfloor^\perp \otimes ?^l \lfloor Bx \rfloor & (\forall_R) & \lceil \forall B \rceil^\perp \otimes \forall x ?^r \lceil Bx \rceil \\
(\exists_L) & \lfloor \exists B \rfloor^\perp \otimes \forall x ?^l \lfloor Bx \rfloor & (\exists_R) & \lceil \exists B \rceil^\perp \otimes ?^r \lceil Bx \rceil \\
(\bot_L) & \lfloor \bot \rfloor^\perp & &
\end{array}
$$

Figure 6.2: The theory, $\mathcal{L}_{g1}$, for the *G1* systems.

$$
\begin{array}{llll}
(\supset_L) & \lfloor A \supset B \rfloor^\perp \otimes (!^l ?^r \lceil A \rceil \otimes ?^l \lfloor B \rfloor) & (\supset_R) & \lceil A \supset B \rceil^\perp \otimes (?^l \lfloor A \rfloor \parr ?^r \lceil B \rceil) \\
(Id'_2) & ?^l \lfloor B \rfloor \otimes !^l ?^r \lceil B \rceil & &
\end{array}
$$

Figure 6.3: Formulas for encoding the minimal and intuitionistic systems of *G1*.

$$
\begin{array}{llllll}
(Id_1) & \lfloor B \rfloor^\perp \otimes \lceil B \rceil^\perp & (Id_2) & ?^l \lfloor B \rfloor \otimes ?^r \lceil B \rceil & (W_R) & \lceil B \rceil^\perp \otimes \bot \\
(C_L) & \lfloor B \rfloor^\perp \otimes (?^l \lfloor B \rfloor \parr ?^l \lfloor B \rfloor) & (C_R) & \lceil B \rceil^\perp \otimes (?^r \lceil B \rceil \parr ?^r \lceil B \rceil) & (W_L) & \lfloor B \rfloor^\perp \otimes \bot
\end{array}
$$

Figure 6.4: Identity rules (cut and initial) and the structural rules (weakening and contraction).

Proposition 6.1 *Let all meta-level atoms, $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$, be assigned negative polarity, let $\Gamma \cup \Delta \cup \{C\}$ be a set of object logic formulas, and let the subexponentials, l and r, be specified by the signature $\langle \{\infty, l, r\}; \{l \preceq \infty, r \preceq \infty\}; \{\infty\}; \{\infty\} \rangle$. Then*

1. $\vdash_{sellf} \mathcal{L}_{g1m} \stackrel{\cdot}{l} \lfloor \Gamma \rfloor \stackrel{\cdot}{r} \lceil C \rceil : \cdot \Uparrow$ *iff* $\Gamma \vdash_{g1m} C$;
2. $\vdash_{sellf} \mathcal{L}_{g1i} \stackrel{\cdot}{l} \lfloor \Gamma \rfloor \stackrel{\cdot}{r} \lceil C \rceil : \cdot \Uparrow$ *iff* $\Gamma \vdash_{g1i} C$;
3. $\vdash_{sellf} \mathcal{L}_{g1c} \stackrel{\cdot}{l} \lfloor \Gamma \rfloor \stackrel{\cdot}{r} \lceil \Delta \rceil : \cdot \Uparrow$ *iff* $\Gamma \vdash_{g1c} \Delta$.

Proof We only show the case for minimal logic, as the other cases follow the same reasoning. We prove both directions by induction on the height of proofs. The base cases are when the proof ends with an instance of the initial rule or the $\bot_L$ rule. We only show the former case:

- Initial Rule:

$$
\frac{}{A \vdash A}\,[\mathrm{I}] \qquad \leftrightsquigarrow \qquad
\dfrac{\dfrac{\dfrac{}{\vdash \mathcal{L}_{g1m} \stackrel{\cdot}{l} \lfloor A \rfloor \stackrel{\cdot}{r} \cdot : \cdot \Downarrow \lfloor A \rfloor^\perp}\,[I_l] \quad \dfrac{}{\vdash \mathcal{L}_{g1m} \stackrel{\cdot}{l} \cdot \stackrel{\cdot}{r} \lceil A \rceil : \cdot \Downarrow \lceil A \rceil^\perp}\,[I_r]}{\vdash \mathcal{L}_{g1m} \stackrel{\cdot}{l} \lfloor A \rfloor \stackrel{\cdot}{r} \lceil A \rceil : \cdot \Downarrow \lfloor A \rfloor^\perp \otimes \lceil A \rceil^\perp}\,[\otimes]}{\vdash \mathcal{L}_{g1m} \stackrel{\cdot}{l} \lfloor A \rfloor \stackrel{\cdot}{r} \lceil A \rceil : \cdot \Uparrow}\,[D_\infty, \exists]
$$

- Cut Rule:

$$
\frac{\Gamma_1 \vdash A \quad \Gamma_2, A \vdash C}{\Gamma_1, \Gamma_2 \vdash C}\,[\mathrm{Cut}] \qquad \leftrightsquigarrow
$$

$$
\dfrac{\dfrac{\dfrac{\vdash \mathcal{L}_{g1m} \stackrel{\cdot}{l} \lfloor \Gamma_2, A \rfloor \stackrel{\cdot}{r} \lceil C \rceil : \cdot \Uparrow}{\vdash \mathcal{L}_{g1m} \stackrel{\cdot}{l} \lfloor \Gamma_2 \rfloor \stackrel{\cdot}{r} \lceil C \rceil : \cdot \Downarrow ?^l \lfloor A \rfloor}\,[R\Downarrow, ?^l] \quad \dfrac{\vdash \mathcal{L}_{g1m} \stackrel{\cdot}{l} \lfloor \Gamma_1 \rfloor \stackrel{\cdot}{r} \lceil A \rceil : \cdot \Uparrow}{\vdash \mathcal{L}_{g1m} \stackrel{\cdot}{l} \lfloor \Gamma_1 \rfloor \stackrel{\cdot}{r} \cdot : \cdot \Downarrow !^l ?^r \lceil A \rceil}\,[!^l, ?^r]}{\vdash \mathcal{L}_{g1m} \stackrel{\cdot}{l} \lfloor \Gamma_1, \Gamma_2 \rfloor \stackrel{\cdot}{r} \lceil C \rceil : \cdot \Downarrow ?^l \lfloor A \rfloor \otimes !^l ?^r \lceil A \rceil}\,[\otimes]}{\vdash \mathcal{L}_{g1m} \stackrel{\cdot}{l} \lfloor \Gamma_1, \Gamma_2 \rfloor \stackrel{\cdot}{r} \lceil C \rceil : \cdot \Uparrow}\,[D_\infty, \exists]
$$

Notice that, because of the presence of the l-bang in Id'_2, the atom $\lceil C \rceil$ must move to the left branch.

• Structural Rules: The cases for the structural rules are straightforward and we do not show them here.

• $\wedge L_1$:

$$\frac{\Gamma, A \vdash C}{\Gamma, A \wedge B \vdash C}\ [\wedge L_1] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{}{\vdash \mathcal{L}_{g1i}\ \dot{l}\ \lfloor A \wedge B \rfloor\ \dot{r}\ \cdot : \cdot \Downarrow \lfloor A \wedge B \rfloor^{\perp}}\ [I_l] \quad \dfrac{\dfrac{\vdash \mathcal{L}_{g1i}\ \dot{l}\ \lfloor \Gamma, A \rfloor\ \dot{r}\ \lceil C \rceil : \cdot \Uparrow}{\vdash \mathcal{L}_{g1i}\ \dot{l}\ \lfloor \Gamma \rfloor\ \dot{r}\ \cdot : \lceil C \rceil \Downarrow ?^{l} \lfloor A \rfloor}\ [R\Downarrow, ?^{l}]}{\vdash \mathcal{L}_{g1i}\ \dot{l}\ \lfloor \Gamma \rfloor\ \dot{r}\ \lceil C \rceil : \cdot \Downarrow ?^{l} \lfloor A \rfloor \oplus ?^{l} \lfloor B \rfloor}\ [\oplus_1]}{\vdash \mathcal{L}_{g1i}\ \dot{l}\ \lfloor \Gamma, A \wedge B \rfloor\ \dot{r}\ \lceil C \rceil : \cdot \Downarrow \lfloor A \wedge B \rfloor^{\perp} \otimes (?^{l} \lfloor A \rfloor \oplus ?^{l} \lfloor B \rfloor)}\ [\otimes]$$

$$\frac{\vdash \mathcal{L}_{g1i}\ \dot{l}\ \lfloor \Gamma, A \wedge B \rfloor\ \dot{r}\ \lceil C \rceil : \cdot \Downarrow \lfloor A \wedge B \rfloor^{\perp} \otimes (?^{l} \lfloor A \rfloor \oplus ?^{l} \lfloor B \rfloor)}{\vdash \mathcal{L}_{g1i}\ \dot{l}\ \lfloor \Gamma, A \wedge B \rfloor\ \dot{r}\ \lceil C \rceil : \cdot \Uparrow}\ [D_{\infty}, 2 \times \exists]$$

The case for the rule $\wedge L_2$ is similar to the derivation above for $\wedge L_1$, but, instead of using an instance of $\oplus_1$ rule, we use an instance of $\oplus_2$ rule.

• $\wedge R$:

$$\frac{\Gamma_1 \vdash A \quad \Gamma_2 \vdash B}{\Gamma_1, \Gamma_2 \vdash A \wedge B}\ [\wedge R] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{}{\vdash \Sigma \Downarrow \lceil A \wedge B \rceil^{\perp}}\ [I_r] \quad \dfrac{\vdash \mathcal{L}_{g1i}\ \dot{l}\ \lfloor \Gamma_1 \rfloor\ \dot{r}\ \lceil A \rceil : \cdot \Uparrow}{\vdash \mathcal{L}_{g1i}\ \dot{l}\ \lfloor \Gamma_1 \rfloor\ \dot{r}\ \cdot : \cdot \Downarrow ?^{r} \lceil A \rceil}\ [R\Downarrow, ?^{r}] \quad \dfrac{\vdash \mathcal{L}_{g1i}\ \dot{l}\ \lfloor \Gamma_2 \rfloor\ \dot{r}\ \lceil B \rceil : \cdot \Uparrow}{\vdash \mathcal{L}_{g1i}\ \dot{l}\ \lfloor \Gamma_2 \rfloor\ \dot{r}\ \cdot : \cdot \Downarrow ?^{r} \lceil B \rceil}\ [R\Downarrow, ?^{r}]}{\vdash \mathcal{L}_{g1i}\ \dot{l}\ \lfloor \Gamma_1, \Gamma_2 \rfloor\ \dot{r}\ \lceil A \wedge B \rceil : \cdot \Downarrow \lceil A \wedge B \rceil^{\perp} \otimes (?^{r} \lceil A \rceil \otimes ?^{r} \lceil B \rceil)}\ [2 \times \otimes]$$

$$\frac{\vdash \mathcal{L}_{g1i}\ \dot{l}\ \lfloor \Gamma_1, \Gamma_2 \rfloor\ \dot{r}\ \lceil A \wedge B \rceil : \cdot \Downarrow \lceil A \wedge B \rceil^{\perp} \otimes (?^{r} \lceil A \rceil \otimes ?^{r} \lceil B \rceil)}{\vdash \mathcal{L}_{g1i}\ \dot{l}\ \lfloor \Gamma_1, \Gamma_2 \rfloor\ \dot{r}\ \lceil A \wedge B \rceil : \cdot \Uparrow}\ [D_{\infty}, 2 \times \exists]$$

where Σ is the set of formulas $\mathcal{L}_{g1i}\ \dot{l}\ \cdot\ \dot{r}\ \lceil A \wedge B \rceil : \cdot$.

The cases for the disjunction left and right introduction rules are similar to the cases for the conjunction introduction rules shown above.

• $\supset R$:

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B}\ [\supset R] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{}{\vdash \mathcal{L}_{g1i}\ \dot{l}\ \cdot\ \dot{r}\ \lceil A \supset B \rceil : \cdot \Downarrow \lceil A \supset B \rceil^{\perp}}\ [I_r] \quad \dfrac{\dfrac{\vdash \mathcal{L}_{g1i}\ \dot{l}\ \lfloor \Gamma, A \rfloor\ \dot{r}\ \lceil B \rceil : \cdot \Uparrow}{\vdash \mathcal{L}_{g1i}\ \dot{l}\ \lfloor \Gamma \rfloor\ \dot{r}\ \cdot : \cdot \Uparrow ?^{l} \lfloor A \rfloor, ?^{r} \lceil B \rceil}\ [?^{l}, ?^{r}]}{\vdash \mathcal{L}_{g1i}\ \dot{l}\ \lfloor \Gamma \rfloor\ \dot{r}\ \cdot : \cdot \Downarrow ?^{l} \lfloor A \rfloor \parr ?^{r} \lceil B \rceil}\ [R\Downarrow, \parr]}{\vdash \mathcal{L}_{g1i}\ \dot{l}\ \lfloor \Gamma \rfloor\ \dot{r}\ \lceil A \supset B \rceil : \cdot \Downarrow \lceil A \supset B \rceil^{\perp} \otimes (?^{l} \lfloor A \rfloor \parr ?^{r} \lceil B \rceil)}\ [\otimes]$$

$$\frac{\vdash \mathcal{L}_{g1i}\ \dot{l}\ \lfloor \Gamma \rfloor\ \dot{r}\ \lceil A \supset B \rceil : \cdot \Downarrow \lceil A \supset B \rceil^{\perp} \otimes (?^{l} \lfloor A \rfloor \parr ?^{r} \lceil B \rceil)}{\vdash \mathcal{L}_{g1i}\ \dot{l}\ \lfloor \Gamma \rfloor\ \dot{r}\ \lceil A \supset B \rceil : \cdot \Uparrow}\ [D_{\infty}, 2 \times \exists]$$

• $\supset L$:

$$\frac{\Gamma_1 \vdash A \quad \Gamma_2, B \vdash C}{\Gamma_1, \Gamma_2, A \supset B \vdash C}\ [\supset L] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{}{\vdash \Sigma \Downarrow \lceil A \supset B \rceil^{\perp}}\,[I_r] \quad \dfrac{\vdash \mathcal{L}_{g1i} \mathbin{\dot{i}} \lfloor \Gamma_1 \rfloor \mathbin{\dot{r}} \lceil A \rceil : \cdot \Uparrow}{\vdash \mathcal{L}_{g1i} \mathbin{\dot{i}} \lfloor \Gamma_1 \rfloor \mathbin{\dot{r}} \cdot : \cdot \Downarrow !^l ?^r \lceil A \rceil}\,[!^l, ?^r] \quad \dfrac{\vdash \mathcal{L}_{g1i} \mathbin{\dot{i}} \lfloor \Gamma_2, B \rfloor \mathbin{\dot{r}} \lceil C \rceil : \cdot \Uparrow}{\vdash \mathcal{L}_{g1i} \mathbin{\dot{i}} \lfloor \Gamma_2 \rfloor \mathbin{\dot{r}} \lceil C \rceil : \cdot \Downarrow ?^l \lfloor B \rfloor}\,[R\Downarrow, ?^l]}{\dfrac{\vdash \mathcal{L}_{g1i} \mathbin{\dot{i}} \lfloor \Gamma_1, \Gamma_2, A \supset B \rfloor \mathbin{\dot{r}} \lceil C \rceil : \cdot \Downarrow \lfloor A \supset B \rfloor^{\perp} \otimes (!^l ?^r \lceil A \rceil \otimes ?^l \lfloor B \rfloor)}{\vdash \mathcal{L}_{g1i} \mathbin{\dot{i}} \lfloor \Gamma_1, \Gamma_2, A \supset B \rfloor \mathbin{\dot{r}} \lceil C \rceil : \cdot \Uparrow}\,[D_\infty, 2 \times \exists]}\,[2 \times \otimes]$$

where Σ is the set of formulas $\mathcal{L}_{g1i} \mathbin{\dot{i}} \lfloor A \supset B \rfloor \mathbin{\dot{r}} \cdot : \cdot$. Again, because of the presence of the subexponential bang in the formula $(\supset_L)$, the formula $\lceil C \rceil$ must go to right-most branch, obtaining, hence, stronger levels of adequacy for the encodings of intuitionistic and minimal logic, *i.e.*, full completeness of derivations.

Finally, the cases for the quantifiers are straightforward. Here we show only the cases for the right universal and existential quantifiers introduction.

- $\forall R$:

$$\dfrac{\Gamma \vdash B\{c/x\}}{\Gamma \vdash \forall x\, B}\,[\forall R] \quad \leftrightsquigarrow \quad \dfrac{\dfrac{}{\vdash \mathcal{L}_{g1i} \mathbin{\dot{i}} \cdot \mathbin{\dot{r}} \lceil \forall B \rceil : \cdot \Downarrow \lceil \forall B \rceil^{\perp}}\,[I_r] \quad \dfrac{\dfrac{\vdash \mathcal{L}_{g1i} \mathbin{\dot{i}} \lfloor \Gamma \rfloor \mathbin{\dot{r}} \lceil Bc \rceil : \cdot \Uparrow}{\vdash \mathcal{L}_{g1i} \mathbin{\dot{i}} \lfloor \Gamma \rfloor \mathbin{\dot{r}} \cdot : \cdot \Uparrow \forall x ?^r \lceil Bx \rceil}\,[\forall, ?^r]}{\vdash \mathcal{L}_{g1i} \mathbin{\dot{i}} \lfloor \Gamma \rfloor \mathbin{\dot{r}} \cdot : \cdot \Downarrow \forall x ?^r \lceil Bx \rceil}\,[R\Downarrow]}{\dfrac{\vdash \mathcal{L}_{g1i} \mathbin{\dot{i}} \lfloor \Gamma \rfloor \mathbin{\dot{r}} \lceil \forall B \rceil : \cdot \Downarrow \lceil \forall B \rceil^{\perp} \otimes \forall x ?^r \lceil Bx \rceil}{\vdash \mathcal{L}_{g1i} \mathbin{\dot{i}} \lfloor \Gamma \rfloor \mathbin{\dot{r}} \lceil \forall B \rceil : \cdot \Uparrow}\,[D_\infty, \exists]}\,[\otimes]$$

Notice that the variable x is instantiated in the meta-logic derivation as an eigenvariable, similarly as in the object logic rule.

- $\exists R$:

$$\dfrac{\Gamma \vdash B\{t/x\}}{\Gamma \vdash \exists x\, B}\,[\exists R] \quad \leftrightsquigarrow \quad \dfrac{\dfrac{}{\vdash \mathcal{L}_{g1i} \mathbin{\dot{i}} \cdot \mathbin{\dot{r}} \lceil \exists B \rceil : \cdot \Downarrow \lceil \exists B \rceil^{\perp}}\,[I_r] \quad \dfrac{\vdash \mathcal{L}_{g1i} \mathbin{\dot{i}} \lfloor \Gamma \rfloor \mathbin{\dot{r}} \lceil Bt \rceil : \cdot \Uparrow}{\vdash \mathcal{L}_{g1i} \mathbin{\dot{i}} \lfloor \Gamma \rfloor \mathbin{\dot{r}} \cdot : \cdot \Downarrow ?^r \lceil Bt \rceil}\,[R\Downarrow, ?^r]}{\dfrac{\vdash \mathcal{L}_{g1i} \mathbin{\dot{i}} \lfloor \Gamma \rfloor \mathbin{\dot{r}} \lceil \exists B \rceil : \cdot \Downarrow \lceil \exists B \rceil^{\perp} \otimes ?^r \lceil Bt \rceil}{\vdash \mathcal{L}_{g1i} \mathbin{\dot{i}} \lfloor \Gamma \rfloor \mathbin{\dot{r}} \lceil \exists B \rceil : \cdot \Uparrow}\,[D_\infty, 2 \times \exists]}\,[\otimes]$$

□

6.4 Multi-conclusion Proof System for Intuitionistic Logic

The system *mLJ*, depicted in Figure 6.5, is a proof system for intuitionistic logic, proposed by Maehara [Maehara 1954], that, differently from the Gentzen system shown before, allows the right-hand-side of sequents to have more than one formula. We use here the additive version of this system, where the context are shared, and denote provability in this system by the judgment $\vdash_{mlj}$. The theory $\mathcal{L}_{mlj}$ in Figure 6.6 encodes *mLJ* and uses two unrelated and unbounded subexponentials, l and r, where the $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ atoms are stored. We obtain the adequacy on the level of derivations by assigning negative polarity to all meta-level atoms, as states the following proposition.

Proposition 6.2 *Let all meta-level atoms, $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$, be assigned negative polarity, let $\Gamma \cup \Delta$ be a set of object logic formulas, and let the subexponentials, l and r, be specified by the signature $\langle \{\infty, l, r\}; \{l \preceq \infty, r \preceq \infty\} \{\infty, l, r\}; \{\infty, l, r\} \rangle$. Then*

$$\vdash_{sellf} \mathcal{L}_{mlj} \mathbin{\dot{i}} \lfloor \Gamma \rfloor \mathbin{\dot{r}} \lceil \Delta \rceil : \cdot \Uparrow \quad \textit{iff } \Gamma \vdash_{mlj} \Delta.$$

$$\frac{\Gamma, A \supset B \vdash A, \Delta \quad \Gamma, A \supset B, B \vdash \Delta}{\Gamma, A \supset B \vdash \Delta}\ [\supset_l] \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B, \Delta}\ [\supset_r]$$

$$\frac{\Gamma, A \wedge B, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\ [\wedge_l] \qquad \frac{\Gamma \vdash A \wedge B, A, \Delta \quad \Gamma \vdash A \wedge B, B, \Delta}{\Gamma \vdash A \wedge B, \Delta}\ [\wedge_r]$$

$$\frac{\Gamma, A \vee B, A, \vdash \Delta \quad \Gamma, A \vee B, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta}\ [\vee_l] \qquad \frac{\Gamma \vdash A \vee B, A, B, \Delta}{\Gamma \vdash A \vee B, \Delta}\ [\vee_r]$$

$$\frac{\Gamma, \forall x\, A, A\{t/x\} \vdash \Delta}{\Gamma, \forall x\, A \vdash \Delta}\ [\forall_l] \qquad \frac{\Gamma \vdash A\{c/x\}}{\Gamma \vdash \Delta, \forall x\, A}\ [\forall_r]$$

$$\frac{\Gamma, \exists x\, A, A\{c/x\} \vdash \Delta}{\Gamma, \exists x\, A \vdash \Delta}\ [\exists_l] \qquad \frac{\Gamma \vdash \Delta, \exists x\, A, A\{t/x\}}{\Gamma \vdash \Delta, \exists x\, A}\ [\exists_r]$$

$$\frac{}{\Gamma, A \vdash A, \Delta}\ [\mathrm{I}] \qquad \frac{\Gamma \vdash B, \Delta \quad \Gamma, B \vdash \Delta}{\Gamma \vdash \Delta}\ [\mathrm{Cut}] \qquad \frac{}{\Gamma, \bot \vdash \Delta}\ [\bot_l]$$

Figure 6.5: The multi-conclusion intuitionistic sequent calculus, mLJ, with additive rules.

$(\supset_l)$	$\lfloor A \supset B \rfloor^\perp \otimes (?^r \lceil A \rceil \,\&\, ?^l \lfloor A \rfloor)$	$(\supset_r)$	$\lceil A \supset B \rceil^\perp \otimes !^l (?^l \lfloor A \rfloor \parr ?^r \lceil B \rceil)$
$(\wedge_l)$	$\lfloor A \wedge B \rfloor^\perp \otimes (?^l \lfloor A \rfloor \parr ?^l \lfloor B \rfloor)$	$(\wedge_r)$	$\lceil A \wedge B \rceil^\perp \otimes (?^r \lceil A \rceil \,\&\, ?^r \lceil B \rceil)$
$(\vee_l)$	$\lfloor A \vee B \rfloor^\perp \otimes (?^l \lfloor A \rfloor \,\&\, ?^l \lfloor B \rfloor)$	$(\vee_r)$	$\lceil A \vee B \rceil^\perp \otimes (?^r \lceil A \rceil \parr ?^r \lceil B \rceil)$
$(\forall_L)$	$\lfloor \forall B \rfloor^\perp \otimes ?^l \lfloor Bx \rfloor$	$(\forall_R)$	$\lceil \forall B \rceil^\perp \otimes !^l \forall x ?^r \lceil Bx \rceil$
$(\exists_L)$	$\lfloor \exists B \rfloor^\perp \otimes \forall x ?^l \lfloor Bx \rfloor$	$(\exists_R)$	$\lceil \exists B \rceil^\perp \otimes ?^r \lceil Bx \rceil$
$(\bot_L)$	$\lfloor \bot \rfloor^\perp$		
(Id_1)	$\lfloor B \rfloor^\perp \otimes \lceil B \rceil^\perp$	(Id_2)	$?^l \lfloor B \rfloor \otimes ?^r \lceil B \rceil$

Figure 6.6: The theory, $\mathcal{L}_{mlj}$, for the multi-conclusion intuitionistic logic system mLJ.

Proof For both direction, we prove by induction on the height of proofs.

One of the base cases is when a proof finishes with an initial rule:

- Initial Rule:

$$\frac{}{\Gamma, A \vdash A, \Delta}\,[\mathrm{I}] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{}{\vdash \mathcal{L}_{mlj}\ \dot{l}\ \lfloor \Gamma, A\rfloor\ \dot{;}\ \lceil \Delta, A\rceil : \cdot \Downarrow \lfloor A\rfloor^{\perp}}\,[I_l] \quad \dfrac{}{\vdash \mathcal{L}_{mlj}\ \dot{l}\ \lfloor \Gamma, A\rfloor\ \dot{;}\ \lceil \Delta, A\rceil : \cdot \Downarrow \lceil A\rceil^{\perp}}\,[I_r]}{\dfrac{\vdash \mathcal{L}_{mlj}\ \dot{l}\ \lfloor \Gamma, A\rfloor\ \dot{;}\ \lceil \Delta, A\rceil : \cdot \Downarrow \lfloor A\rfloor^{\perp} \otimes \lceil A\rceil^{\perp}}{\vdash \mathcal{L}_{mlj}\ \dot{l}\ \lfloor \Gamma, A\rfloor\ \dot{;}\ \lceil \Delta, A\rceil : \cdot \Uparrow}\,[D_\infty, \exists]}\,[\otimes]$$

Since all $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ are assigned negative polarity, it must be case that both atoms $\lfloor A\rfloor$ and $\lceil A\rceil$ are present in the context whenever one focuses on the Id_1 clause.

The other base case is when the proof ends with $\perp_l$,

- $\perp_l$:

$$\frac{}{\Gamma, \perp \vdash \Delta}\,[\perp_l] \qquad \leftrightsquigarrow \qquad \dfrac{\dfrac{}{\vdash \mathcal{L}_{mlj}\ \dot{l}\ \lfloor \Gamma, \perp\rfloor\ \dot{;}\ \lceil \Delta\rceil : \cdot \Downarrow \lfloor \perp\rfloor^{\perp}}\,[I_r]}{\vdash \mathcal{L}_{mlj}\ \dot{l}\ \lfloor \Gamma, \perp\rfloor\ \dot{;}\ \lceil \Delta\rceil : \cdot \Uparrow}\,[D_\infty]$$

We only show the interesting inductive cases, as most of them follow the same reasoning as in the proof of Proposition 6.1. We start with the case for the right implication introduction rule:

- $\supset_r$:

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B, \Delta}\,[\supset_r] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{}{\vdash \Sigma \Downarrow \lceil A \supset B\rceil^{\perp}}\,[I_r] \quad \dfrac{\dfrac{\vdash \mathcal{L}_{mlj}\ \dot{l}\ \lfloor \Gamma, A\rfloor\ \dot{;}\ \lceil B\rceil : \cdot \Uparrow}{\vdash \mathcal{L}_{mlj}\ \dot{l}\ \lfloor \Gamma\rfloor\ \dot{;}\ \cdot : \cdot \Uparrow ?^{l}\lfloor A\rfloor \parr ?^{r}\lceil B\rceil}\,[\parr, ?^{l}, ?^{r}]}{\vdash \mathcal{L}_{mlj}\ \dot{l}\ \lfloor \Gamma\rfloor\ \dot{;}\ \lceil \Delta, A \supset B\rceil : \cdot \Downarrow !^{l}(?^{l}\lfloor A\rfloor \parr ?^{r}\lceil B\rceil)}\,[!^{l}]}{\dfrac{\vdash \mathcal{L}_{mlj}\ \dot{l}\ \lfloor \Gamma\rfloor\ \dot{;}\ \lceil \Delta, A \supset B\rceil : \cdot \Downarrow \lceil A \supset B\rceil^{\perp} \otimes !_l(?_l\lfloor A\rfloor \parr ?_r\lceil B\rceil)}{\vdash \mathcal{L}_{mlj}\ \dot{l}\ \lfloor \Gamma\rfloor\ \dot{;}\ \lceil \Delta, A \supset B\rceil : \cdot \Uparrow}\,[D_\infty, 2 \times \exists]}\,[\otimes]$$

where Σ is the set of formulas $\mathcal{L}_{mlj}\ \dot{l}\ \lfloor \Gamma\rfloor\ \dot{;}\ \lceil \Delta, A \supset B\rceil : \cdot$. The role of the subexponential bang in the clause $(\supset_R)$ is crucial for the correct encoding of the object-logic rule, as it enforces that the $\lceil\cdot\rceil$ formulas in the context to be weakened before the atoms $\lfloor A\rfloor$ and $\lceil B\rceil$ are moved to the context.

Another interesting case is the universal quantifier right introduction rule:

- $\forall_r$:

$$\frac{\Gamma \vdash A\{c/x\}}{\Gamma \vdash \Delta, \forall x\, A}\,[\forall_r] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{}{\vdash \mathcal{L}_{mlj}\ \dot{l}\ \lfloor \Gamma\rfloor\ \dot{;}\ \lceil \Delta, \forall x\, A\rceil : \cdot \Downarrow \lceil \forall x\, A\rceil^{\perp}}\,[I_r] \quad \dfrac{\dfrac{\vdash \mathcal{L}_{mlj}\ \dot{l}\ \lfloor \Gamma\rfloor\ \dot{;}\ \lceil Ac\rceil : \cdot \Uparrow}{\vdash \mathcal{L}_{mlj}\ \dot{l}\ \lfloor \Gamma\rfloor\ \dot{;}\ \cdot : \cdot \Uparrow \forall x ?^{r}\lceil Ax\rceil}\,[\forall, ?^{r}]}{\vdash \mathcal{L}_{mlj}\ \dot{l}\ \lfloor \Gamma\rfloor\ \dot{;}\ \lceil \Delta, \forall x\, A\rceil : \cdot \Downarrow !^{l}\forall x ?^{r}\lceil Ax\rceil}\,[!^{l}]}{\dfrac{\vdash \mathcal{L}_{mlj}\ \dot{l}\ \lfloor \Gamma\rfloor\ \dot{;}\ \lceil \Delta, \forall x\, A\rceil : \cdot \Downarrow \lceil \forall B\rceil^{\perp} \otimes !^{l}\forall x ?^{r}\lceil Bx\rceil}{\vdash \mathcal{L}_{mlj}\ \dot{l}\ \lfloor \Gamma\rfloor\ \dot{;}\ \lceil \Delta, \forall x\, A\rceil : \cdot \Uparrow}\,[D_\infty, \exists]}\,[\otimes]$$

$$\frac{\Gamma, A \supset B \to A; \cdot \quad \Gamma, A \supset B, B \vdash \Delta}{\Gamma, A \supset B \vdash \Delta}\ [\supset_l] \qquad \frac{\Gamma, A \vdash B}{\Gamma \to A \supset B; \Delta}\ [\supset_r]$$

$$\frac{\Gamma, A \vee B, A \vdash \Delta \quad \Gamma, A \vee B, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta}\ [\vee_l] \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \to A \vee B; \Delta}\ [\vee_r]$$

$$\frac{\Gamma, A \wedge B, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\ [\wedge_l] \qquad \frac{\Gamma \to A; \Delta \quad \Gamma \to B; \Delta}{\Gamma \to A \wedge B; \Delta}\ [\wedge_r]$$

$$\frac{}{\Gamma, A \to A; \Delta}\ [\mathrm{I}] \qquad \frac{\Gamma \to C; \Delta}{\Gamma \vdash C, \Delta}\ [D] \qquad \frac{}{\Gamma, \bot \vdash \Delta}\ [\bot_l]$$

Figure 6.7: The focused multi-conclusion system for intuitionistic logic - LJQ^*.

(Id_1)	$\lfloor A \rfloor^\perp \otimes \lceil A \rceil^\perp$	$(\bot_L)$	$\lfloor \bot \rfloor^\perp$
$(\supset_L)$	$\lfloor A \supset B \rfloor^\perp \otimes (!^l ?^f \lceil A \rceil \otimes !^r ?^l \lfloor B \rfloor)$	$(\supset_R)$	$\lceil A \supset B \rceil^\perp \otimes !^l (?^l \lfloor A \rfloor \parr ?^r \lceil B \rceil)$
$(\vee_L)$	$\lfloor A \vee B \rfloor^\perp \otimes (!^r ?^l \lfloor A \rfloor \otimes !^r ?^l \lfloor B \rfloor)$	$(\vee_R)$	$\lceil A \vee B \rceil^\perp \otimes !^r (?^r \lceil A \rceil \parr ?^r \lceil B \rceil)$
$(\wedge_L)$	$\lfloor A \wedge B \rfloor^\perp \otimes !^r (?^l \lfloor A \rfloor \parr ?^l \lfloor B \rfloor)$	$(\wedge_R)$	$\lceil A \wedge B \rceil^\perp \otimes (!^r ?^f \lceil A \rceil \otimes !^r ?^f \lceil B \rceil)$

Figure 6.8: The theory $\mathcal{L}_{ljq}$ used to encode the system LJQ^*.

As in the previous case, the role of the subexponential bang is to weaken all the $\lceil \cdot \rceil$ formulas in the context before the formula $\lceil Ac \rceil$ is moved to the context. □

6.5 A focused multi-conclusion system for Intuitionistic Logic

In the next sections, we move our attention to focused systems. We start with the focused multi-conclusion system for intuitionistic logic LJQ^*, depicted in Figure 6.7, which is a variant of the system proposed by Herbelin [Herbelin 1995, page 78] and was used by Dyckhoff & Lengrand in [Dyckhoff 2006]. LJQ^* has two types of sequents: unfocused sequents of the form $\Gamma \vdash \Delta$ and focused sequents of the form $\Gamma \to A; \Delta$ where the formula A, in the *stoup*, is focused on. We use the judgment $\Gamma \vdash_{ljq} \Delta$ to denote that the sequent $\Gamma \vdash \Delta$ is provable in LJQ^* and the judgment $\Gamma \to_{ljq} A; \Delta$ to denote that the sequent $\Gamma \to A; \Delta$ is provable in LJQ^*. As is usual in focused systems, the *stoup*, together with the decide rule D, restricts the proofs available by imposing some focusing discipline to proofs. Here proofs are restricted as follows: the logical right introduction rules introduce only focused sequents, while the left introduction rules introduce only unfocused sequents.

The theory depicted in Figure 6.8 uses three subexponentials: l, used to encode the left-hand-side of object-logic sequents; r, used to encode the right-hand-side of object-logic sequents; and f, used to encode the stoup of object-logic focused sequents. As it will become clear in the proof of Proposition 6.3, the restrictions to sequents imposed by the focusing discipline, via the decide rule, are encoded implicitly by the use of the subexponentials. Therefore, the clauses encoding the inference rules that introduce focused sequents will specify two object logic derivations: one derivation is the object-logic inference rule whose conclusion is a focused sequent, and the other derivation is the object-logic derivation composed of a decide rule and the corresponding object-logic inference rule.

Proposition 6.3 *Let all meta-level atoms, $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$, be assigned negative polarity, let $\Gamma \cup \Delta \cup \{C\}$ be a set of object logic formulas, and let the subexponentials l, r and f, be specified by the signature $\langle \{f, l, r, \infty\} \{f \preceq r \preceq l \preceq \infty\}; \{l, r, \infty\}; \{l, r, \infty\} \rangle$. Then*

1. $\vdash_{sellf} \mathcal{L}_{ljq} \stackrel{\cdot}{i} \lfloor \Gamma \rfloor \stackrel{\cdot}{i} \lceil \Delta \rceil \stackrel{\cdot}{i} \cdot : \cdot \Uparrow$ *iff* $\Gamma \vdash_{ljq} \Delta$

2. $\vdash_{sellf} \mathcal{L}_{ljq} \stackrel{i}{} \lfloor \Gamma \rfloor \stackrel{i}{r} \lceil \Delta \rceil \stackrel{i}{f} \lceil C \rceil : \cdot \Uparrow$ *iff* $\Gamma \rightarrow_{ljq} C; \Delta$.

Proof We prove both directions by induction on the height of proofs. The base case is when a proof finishes with an initial rule or a $\bot_l$ rule:

- Initial Rule:

$$\frac{}{\Gamma, A \rightarrow A; \Delta}\ [\mathrm{I}] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{}{\vdash \mathcal{L}_{ljq} \stackrel{i}{} \lfloor \Gamma, A \rfloor \stackrel{i}{r} \lceil \Delta \rceil \stackrel{i}{f} \cdot : \cdot \Downarrow \lfloor A \rfloor^{\perp}}\ [I_l] \quad \dfrac{}{\vdash \mathcal{L}_{ljq} \stackrel{i}{} \lfloor \Gamma, A \rfloor \stackrel{i}{r} \lceil \Delta \rceil \stackrel{i}{f} \lceil A \rceil : \cdot \Downarrow \lceil A \rceil^{\perp}}\ [I_f]}{\vdash \mathcal{L}_{ljq} \stackrel{i}{} \lfloor \Gamma, A \rfloor \stackrel{i}{r} \lceil \Delta \rceil \stackrel{i}{f} \lceil A \rceil : \cdot \Downarrow \lfloor A \rfloor^{\perp} \otimes \lceil A \rceil^{\perp}}\ [\otimes]}{\vdash \mathcal{L}_{ljq} \stackrel{i}{} \lfloor \Gamma, A \rfloor \stackrel{i}{r} \lceil \Delta \rceil \stackrel{i}{f} \lceil A \rceil : \cdot \Uparrow}\ [D_\infty, \exists]$$

As decide rules are incorporated in the theory and not encoded explicitly by clauses in $\mathcal{L}_{ljq}$, there is yet another focused derivation that introduces the formula Id_1 that does not have any formula in the context of the subexponential f, as illustrates the following derivation:

$$\dfrac{\dfrac{}{\Gamma, A \rightarrow A; \Delta}\ [\mathrm{I}]}{\Gamma, A \vdash A, \Delta}\ [D] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{}{\vdash \mathcal{L}_{ljq} \stackrel{i}{} \lfloor \Gamma, A \rfloor \stackrel{i}{r} \lceil \Delta, A \rceil \stackrel{i}{f} \cdot : \cdot \Downarrow \lceil A \rceil^{\perp}}\ [I_f] \quad \dfrac{}{\vdash \mathcal{L}_{ljq} \stackrel{i}{} \lfloor \Gamma, A \rfloor \stackrel{i}{r} \lceil \Delta, A \rceil \stackrel{i}{f} \cdot : \cdot \Downarrow \lfloor A \rfloor^{\perp}}\ [I_l]}{\vdash \mathcal{L}_{ljq} \stackrel{i}{} \lfloor \Gamma, A \rfloor \stackrel{i}{r} \lceil \Delta, A \rceil \stackrel{i}{f} \cdot : \cdot \Downarrow \lfloor A \rfloor^{\perp} \otimes \lceil A \rceil^{\perp}}\ [\otimes]}{\vdash \mathcal{L}_{ljq} \stackrel{i}{} \lfloor \Gamma, A \rfloor \stackrel{i}{r} \lceil \Delta, A \rceil \stackrel{i}{f} \cdot : \cdot \Uparrow}\ [D_\infty, \exists]$$

As the subexponential f does not allow weakening, it must be the case that either $\lceil A \rceil$ is in the context f or it is empty. Thus, satisfying the restrictions on focused proofs that only one formula is focused on.

The other base case is when the proof ends with a $\bot_l$ rule which is similar to the corresponding case in Proposition 6.3.

We now proceed to the inductive cases starting with the left implication introduction rule.

- $\supset_l$:

$$\frac{\Gamma, A \supset B \rightarrow A; \cdot \quad \Gamma, A \supset B, B \vdash \Delta}{\Gamma, A \supset B \vdash \Delta}\ [\supset_l] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{}{\vdash \mathcal{K} \Downarrow \lfloor A \supset B \rfloor^{\perp}}\ [I_l] \quad \dfrac{\dfrac{\vdash \mathcal{L}_{ljq} \stackrel{i}{} \lfloor \Gamma' \rfloor \stackrel{i}{r} \cdot \stackrel{i}{f} \lceil A \rceil : \cdot \Uparrow}{\vdash \mathcal{L}_{ljq} \stackrel{i}{} \lfloor \Gamma' \rfloor \stackrel{i}{r} \cdot \stackrel{i}{f} \cdot : \cdot \Uparrow ?^f \lceil A \rceil}\ [?^f]}{\vdash \mathcal{L}_{ljq} \stackrel{i}{} \lfloor \Gamma' \rfloor \stackrel{i}{r} \lceil \Delta \rceil \stackrel{i}{f} \cdot : \cdot \Downarrow !^l ?^f \lceil A \rceil}\ [!^l] \quad \dfrac{\vdash \mathcal{L}_{ljq} \stackrel{i}{} \lfloor \Gamma', B \rfloor \stackrel{i}{r} \lceil \Delta \rceil \stackrel{i}{f} \cdot : \cdot \Uparrow}{\vdash \mathcal{L}_{ljq} \stackrel{i}{} \lfloor \Gamma' \rfloor \stackrel{i}{r} \lceil \Delta \rceil \stackrel{i}{f} \cdot : \cdot \Downarrow !^r ?^f \lceil B \rceil}\ [!^l, ?^l]}{\vdash \mathcal{L}_{ljq} \stackrel{i}{} \lfloor \Gamma' \rfloor \stackrel{i}{r} \lceil \Delta \rceil \stackrel{i}{f} \cdot : \cdot \Downarrow \lfloor A \supset B \rfloor^{\perp} \otimes (!^l ?^f \lceil A \rceil \otimes !^r ?^l \lfloor B \rfloor)}\ [2 \times \otimes]}{\vdash \mathcal{L}_{ljq} \stackrel{i}{} \lfloor \Gamma' \rfloor \stackrel{i}{r} \lceil \Delta \rceil \stackrel{i}{f} \cdot : \cdot \Uparrow}\ [D_\infty, 2 \times \exists]$$

where $\mathcal{K}$ is the set of formulas $\mathcal{L}_{ljq} \stackrel{i}{} \lfloor \Gamma' \rfloor \stackrel{i}{r} \lceil \Delta \rceil$ and Γ' is the set of formulas $\Gamma \cup \{A \supset B\}$. The role of the subexponentials bangs, $!^l$ and $!^r$, in the clause $(\supset_L)$, is crucial to obtain the correspondence between the meta-logic and object-logic derivations above. The l-bang enforces that the set of atoms, $\lceil \Delta \rceil$, containing object-logic formulas in the right of the sequent, is weakened in the left open premise, while the r-bang ensures that there are no focused atoms in the right open premise, that is, the f-context is empty, and, hence, enforcing the focusing discipline that a formula on the left can be introduced only if no formula is focused on.

• $\supset_r$: The case for the right implication introduction rule, depicted below, follows a similar line of reasoning as used in the previous case; we use a l-bang to weaken the formulas in the context of the subexponential r.

$$\frac{\Gamma, A \vdash B}{\Gamma \to A \supset B; \Delta}\,[\supset_r] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{}{\vdash \Sigma \Downarrow \lceil A \supset B \rceil^{\perp}}\,[I_r] \quad \dfrac{\vdash \mathcal{L}_{ljq} \mathbin{\dot{l}} \lfloor \Gamma, A \rfloor \mathbin{\dot{r}} \lceil B \rceil \mathbin{\dot{f}} \cdot : \cdot \Uparrow}{\vdash \mathcal{L}_{ljq} \mathbin{\dot{l}} \lfloor \Gamma \rfloor \mathbin{\dot{r}} \lceil \Delta \rceil \mathbin{\dot{f}} \cdot : \cdot \Downarrow !^l(?^l \lfloor A \rfloor \parr ?^r \lceil B \rceil)}\,[!^l, \parr, ?^r, ?^l]}{\dfrac{\vdash \mathcal{L}_{ljq} \mathbin{\dot{l}} \lfloor \Gamma \rfloor \mathbin{\dot{r}} \lceil \Delta \rceil \mathbin{\dot{f}} \lceil A \supset B \rceil : \cdot \Downarrow \lceil A \supset B \rceil^{\perp} \otimes !^l(?^l \lfloor A \rfloor \parr ?^r \lceil B \rceil)}{\vdash \mathcal{L}_{ljq} \mathbin{\dot{l}} \lfloor \Gamma \rfloor \mathbin{\dot{r}} \lceil \Delta \rceil \mathbin{\dot{f}} \lceil A \supset B \rceil : \cdot \Uparrow}\,[D_\infty, 2 \times \exists]}\,[\otimes]$$

where Σ is the set of formulas $\mathcal{L}_{ljq} \mathbin{\dot{l}} \lfloor \Gamma \rfloor \mathbin{\dot{r}} \lceil \Delta \rceil \mathbin{\dot{f}} \lceil A \supset B \rceil$.

• $\vee_l$:

$$\frac{\Gamma, A \vee B, A \vdash \Delta \quad \Gamma, A \vee B, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta}\,[\vee_l] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{}{\vdash \mathcal{K} \Downarrow \lfloor A \vee B \rfloor^{\perp}}\,[I_l] \quad \dfrac{\vdash \mathcal{L}_{ljq} \mathbin{\dot{l}} \lfloor \Gamma', A \rfloor \mathbin{\dot{r}} \lceil \Delta \rceil \mathbin{\dot{f}} \cdot : \cdot \Uparrow}{\vdash \mathcal{L}_{ljq} \mathbin{\dot{l}} \lfloor \Gamma' \rfloor \mathbin{\dot{r}} \lceil \Delta \rceil \mathbin{\dot{f}} \cdot : \cdot \Downarrow !^r ?^l \lfloor A \rfloor}\,[!^r, ?^l] \quad \dfrac{\vdash \mathcal{L}_{ljq} \mathbin{\dot{l}} \lfloor \Gamma', B \rfloor \mathbin{\dot{r}} \lceil \Delta \rceil \mathbin{\dot{f}} \cdot : \cdot \Uparrow}{\vdash \mathcal{L}_{ljq} \mathbin{\dot{l}} \lfloor \Gamma' \rfloor \mathbin{\dot{r}} \lceil \Delta \rceil \mathbin{\dot{f}} \cdot : \cdot \Downarrow !^r ?^l \lfloor B \rfloor}\,[!^r, ?^l]}{\dfrac{\vdash \mathcal{L}_{ljq} \mathbin{\dot{l}} \lfloor \Gamma' \rfloor \mathbin{\dot{r}} \lceil \Delta \rceil \mathbin{\dot{f}} \cdot : \cdot \Downarrow \lfloor A \vee B \rfloor^{\perp} \otimes (!^r ?^l \lfloor A \rfloor \otimes !^r ?^l \lfloor B \rfloor)}{\vdash \mathcal{L}_{ljq} \mathbin{\dot{l}} \lfloor \Gamma' \rfloor \mathbin{\dot{r}} \lceil \Delta \rceil \mathbin{\dot{f}} \cdot : \cdot \Uparrow}\,[D_\infty, 2 \times \exists]}\,[2 \times \otimes]$$

where $\mathcal{K}$ is the set of formulas $\mathcal{L}_{ljq} \mathbin{\dot{l}} \lfloor \Gamma' \rfloor \mathbin{\dot{r}} \lceil \Delta \rceil$ and Γ' is the set of object-logic formulas $\Gamma \cup \{A \vee B\}$. Here, we again use r-bangs in the clause $(\vee_L)$ to ensure that there are no formulas in the f subexponential context, and, therefore, it is not the case that we conclude a meta-logic sequent that encodes a focused object-logic sequent.

• $\vee_r$:

$$\frac{\Gamma \vdash A, B, \Delta}{\Gamma \to A \vee B; \Delta}\,[\vee_r] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{}{\vdash \Sigma \Downarrow \lceil A \vee B \rceil^{\perp}}\,[I_f] \quad \dfrac{\vdash \mathcal{L}_{ljq} \mathbin{\dot{l}} \lfloor \Gamma \rfloor \mathbin{\dot{r}} \lceil \Delta, A, B \rceil \mathbin{\dot{f}} \cdot : \cdot \Uparrow}{\vdash \mathcal{L}_{ljq} \mathbin{\dot{l}} \lfloor \Gamma \rfloor \mathbin{\dot{r}} \lceil \Delta \rceil \mathbin{\dot{f}} \cdot : \cdot \Downarrow !^r(?^r \lceil A \rceil \parr ?^r \lceil B \rceil)}\,[!^r, \parr, 2 \times ?]}{\dfrac{\vdash \mathcal{L}_{ljq} \mathbin{\dot{l}} \lfloor \Gamma \rfloor \mathbin{\dot{r}} \lceil \Delta \rceil \mathbin{\dot{f}} \lceil A \wedge B \rceil : \cdot \Downarrow \lceil A \vee B \rceil^{\perp} \otimes !^r(?^r \lceil A \rceil \parr ?^r \lceil B \rceil)}{\vdash \mathcal{L}_{ljq} \mathbin{\dot{l}} \lfloor \Gamma \rfloor \mathbin{\dot{r}} \lceil \Delta \rceil \mathbin{\dot{f}} \lceil A \vee B \rceil : \cdot \Uparrow}\,[D_\infty, 2 \times \exists]}\,[\otimes]$$

where Σ is the set of formulas $\mathcal{L}_{ljq} \mathbin{\dot{l}} \lfloor \Gamma \rfloor \mathbin{\dot{r}} \lceil \Delta \rceil \mathbin{\dot{f}} \lceil A \vee B \rceil$. As before, we use the r-bang to ensure that the formula in the f-context is consumed in the left branch.

• $\wedge_l$:

$$\frac{\Gamma, A \wedge B, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\,[\wedge_l] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{}{\vdash \mathcal{K} \Downarrow \lfloor A \wedge B \rfloor^{\perp}}\,[I_l] \quad \dfrac{\vdash \mathcal{L}_{ljq} \mathbin{\dot{l}} \lfloor \Gamma', A, B \rfloor \mathbin{\dot{r}} \lceil \Delta \rceil \mathbin{\dot{f}} \cdot : \cdot \Uparrow}{\vdash \mathcal{L}_{ljq} \mathbin{\dot{l}} \lfloor \Gamma' \rfloor \mathbin{\dot{r}} \lceil \Delta \rceil \mathbin{\dot{f}} \cdot : \cdot \Downarrow !^r(?^l \lfloor A \rfloor \parr ?^l \lfloor B \rfloor)}\,[!^r, \parr, 2 \times ?^l]}{\dfrac{\vdash \mathcal{L}_{ljq} \mathbin{\dot{l}} \lfloor \Gamma' \rfloor \mathbin{\dot{r}} \lceil \Delta \rceil \mathbin{\dot{f}} \cdot : \cdot \Downarrow \lfloor A \wedge B \rfloor^{\perp} \otimes !^r(?^l \lfloor A \rfloor \parr ?^l \lfloor B \rfloor)}{\vdash \mathcal{L}_{ljq} \mathbin{\dot{l}} \lfloor \Gamma' \rfloor \mathbin{\dot{r}} \lceil \Delta \rceil \mathbin{\dot{f}} \cdot : \cdot \Uparrow}\,[D_\infty, 2 \times \exists]}\,[\otimes]$$

$$\frac{\vdash [\Theta, P_a], \Gamma}{\vdash [\Theta], \Gamma, P_a}\ [\text{[]}] \qquad \frac{\mapsto [\Theta], P}{\vdash [\Theta, P]}\ [D] \qquad \frac{\vdash [\Theta], N}{\mapsto [\Theta], N}\ [R]$$

$$\frac{}{\mapsto [\Theta, A_p^{\perp}], A_p}\ [\text{I}] \qquad \frac{}{\mapsto [\Theta], t}\ [t_1] \qquad \frac{}{\vdash [\Theta], \Gamma, \perp^{\perp}}\ [\perp] \qquad \frac{\vdash [\Theta], \Gamma}{\vdash [\Theta], \Gamma, t^{\perp}}\ [t_2]$$

$$\frac{\vdash [\Theta], \Gamma, A \quad \vdash [\Theta], \Gamma, B}{\vdash [\Theta], \Gamma, A \wedge^{-} B}\ [\wedge^{-}] \qquad \frac{\vdash [\Theta], \Gamma, A, B}{\vdash [\Theta], \Gamma, A \vee^{-} B}\ [\vee^{-}]$$

$$\frac{\mapsto [\Theta], B\{t/x\}}{\mapsto [\Theta], \exists x B}\ [\exists] \qquad \frac{\vdash [\Theta], \Gamma, B\{c/x\}}{\vdash [\Theta], \Gamma, \forall x B}\ [\forall]$$

$$\frac{\mapsto [\Theta], A \quad \mapsto [\Theta], B}{\mapsto [\Theta], A \wedge^{+} B}\ [\wedge^{+}] \qquad \frac{\mapsto [\Theta], A_i}{\mapsto [\Theta], A_1 \vee^{+} A_2}\ [\vee^{+}]$$

Figure 6.9: The rules for *LKF*. Here, A_p is a positive literal, P is a positive formula, N is a negative formula, P_a is a positive formula or a literal, Θ is a multiset of positive formulas or literals, Γ is a multiset of formulas, and in the rule $\forall$, c does not appear free in Θ nor in Γ.

where $\mathcal{K}$ is the set of formulas $\mathcal{L}_{ljq}\ \dot{l}\ \lfloor \Gamma' \rfloor\ \dot{r}\ \lceil \Delta \rceil$ and $\Gamma' = \Gamma \cup \{A \wedge B\}$. Once again the r-bang enforces that the context for the subexponential f is empty, which corresponds to the focusing discipline that a formula on the left is introduced only if no formula is focused on.

- $\wedge_r$:

$$\frac{\Gamma \to A; \Delta \quad \Gamma \to B; \Delta}{\Gamma \to A \wedge B; \Delta}\ [\wedge_r] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{\dfrac{}{\vdash \Sigma \Downarrow \lceil A \wedge B \rceil^{\perp}}\ [I_f] \quad \dfrac{\vdash \mathcal{L}_{ljq}\ \dot{l}\ \lfloor \Gamma \rfloor\ \dot{r}\ \lceil \Delta \rceil\ \dot{f}\ \lceil A \rceil : \cdot \Uparrow}{\vdash \mathcal{L}_{ljq}\ \dot{l}\ \lfloor \Gamma \rfloor\ \dot{r}\ \lceil \Delta \rceil\ \dot{f} \cdot : \cdot \Downarrow !^r ?^f \lceil A \rceil}\ [!^r, ?^f] \quad \dfrac{\vdash \mathcal{L}_{ljq}\ \dot{l}\ \lfloor \Gamma \rfloor\ \dot{r}\ \lceil \Delta \rceil\ \dot{f}\ \lceil B \rceil : \cdot \Uparrow}{\vdash \mathcal{L}_{ljq}\ \dot{l}\ \lfloor \Gamma \rfloor\ \dot{r}\ \lceil \Delta \rceil\ \dot{f} \cdot : \cdot \Downarrow !^r ?^f \lceil B \rceil}\ [!^r, ?^f]}{\vdash \mathcal{L}_{ljq}\ \dot{l}\ \lfloor \Gamma \rfloor\ \dot{r}\ \lceil \Delta \rceil\ \dot{f}\ \lceil A \wedge B \rceil : \cdot \Downarrow \lceil A \wedge B \rceil^{\perp} \otimes (!^r ?^f \lceil A \rceil \otimes !^r ?^f \lceil B \rceil)}\ [2 \times \otimes]}{\vdash \mathcal{L}_{ljq}\ \dot{l}\ \lfloor \Gamma \rfloor\ \dot{r}\ \lceil \Delta \rceil\ \dot{f}\ \lceil A \wedge B \rceil : \cdot \Uparrow}\ [D_\infty, 2 \times \exists]$$

where Σ is the set of formulas $\mathcal{L}_{ljq}\ \dot{l}\ \lfloor \Gamma \rfloor\ \dot{r}\ \lceil \Delta \rceil\ \dot{f}\ \lceil A \wedge B \rceil$. The use of the r-bangs has the same motivation as in the clause $(\vee_R)$, namely to ensure that there is at most one formula in the f-context. □

6.6 LKF

Liang & Miller proposed in [Liang 2008, Liang 2007] the system *LKF* for classical logic whose fragment is depicted in Figure 6.9, which is similar to the focused system *LJF* for intuitionistic logic that we described in Chapter 3. As in *LJF*, we assign arbitrarily the polarity for literals and classify as positive the formulas whose main connective is either $\wedge^{+}$, $\vee^{+}$, $\exists$ or t and the positive literals. The remaining formulas are classified as negative. This classification is specified by the definitions *pos*, *syn*, *asyn*, *lit*, and *litSyn* depicted in Figure 6.11. Moreover, literals are specified by using the terms lit^p and lit^n, whose superscripts denote the polarity of a literal: p for positive polarity literals and n for negative polarity literals. We restrict ourselves to the fragment without implications to avoid computing the negation normal form of object-logic formulas.

The system *LKF* has three types of sequents: $\mapsto [\Theta], P$ is the sequent focused on the formula P; $\vdash [\Theta], \Delta$ is the unfocused sequent; $\vdash [\Theta]$ is the sequent that does not contain any asynchronous

$$
\begin{array}{llll}
(\wedge^+) & \lceil A \wedge^+ B \rceil^\perp \otimes (str^s\ A \otimes str^s\ B) & (\wedge^-) & \lceil A \wedge^- B \rceil^\perp \otimes (str^a_{\&}\ A\ B) \\
(\vee^+) & \lceil A_1 \vee^+ A_2 \rceil^\perp \otimes (str^s\ A_1 \oplus str^s\ A_2) & (\vee^-) & \lceil A \vee^- B \rceil^\perp \otimes (str^a_{\parr}\ A\ B) \\
(\exists) & \lceil \exists B \rceil^\perp \otimes (str^s\ Bx) & (\forall) & \lceil \forall B \rceil^\perp \otimes (str^a\ B) \\
(t_1) & \lceil t \rceil^\perp \otimes \top & (t_2) & \lceil t^\perp \rceil^\perp \otimes \perp \\
(\perp) & \lceil \perp^\perp \rceil^\perp \otimes \top & (Id_1) & \lceil lit^p\ A \rceil^\perp \otimes \lceil lit^n\ A^\perp \rceil^\perp
\end{array}
$$

Figure 6.10: The theory $\mathcal{L}_{lkf}$ used to encode *LKF*.

$$
\begin{array}{lcl}
syn\ A & \triangleq & \exists A_1 A_2 (A = A_1 \wedge^+ A_2) \oplus \exists A_1 A_2 (A = A_1 \vee^+ A_2) \oplus \exists B (A = \exists B) \oplus (A = t) \\
asyn\ A & \triangleq & \exists A_1 A_2 (A = A_1 \wedge^- A_2) \oplus \exists A_1 A_2 (A = A_1 \vee^- A_2) \oplus \exists B (A = \forall B) \oplus (A = \perp) \\
lit\ A & \triangleq & \exists A_1 [(A = lit^n\ A_1) \oplus (A = lit^p\ A_1)] \\
litSyn\ A & \triangleq & \exists A_1 [(lit\ A_1) \oplus (syn\ A_1)] \\
pos\ A & \triangleq & \exists A_1 [(A = lit^p\ A_1) \oplus (syn\ A_1)] \\
\\
str^a_\bullet\ A\ B & \triangleq & [(litSyn\ A) \otimes (litSyn\ B) \otimes (?^s \lceil A \rceil \bullet ?^s \lceil B \rceil)] \oplus \\
 & & [(litSyn\ A) \otimes (asyn\ B) \otimes (?^s \lceil A \rceil \bullet \lceil B \rceil)] \oplus \\
 & & [(asyn\ A) \otimes (litSyn\ B) \otimes (\lceil A \rceil \bullet ?^s \lceil B \rceil)] \oplus \\
 & & [(asyn\ A) \otimes (asyn\ B) \otimes (\lceil A \rceil \bullet \lceil B \rceil)] \\
\\
str^a\ A & \triangleq & \exists B (A = \lambda x B) \otimes \{[litSyn\ B \otimes (\forall x ?^s \lceil Ax \rceil)] \oplus [asyn\ B \otimes (\forall x \lceil Ax \rceil)]\} \\
\\
str^s\ A & \triangleq & [(pos\ A) \otimes (!^s\ ?^f \lceil A \rceil)] \oplus [(asyn\ A) \otimes (!^s \lceil A \rceil)] \oplus [\exists A_1 (A = lit^n\ A_1) \otimes (!^s ?^s \lceil A \rceil)]
\end{array}
$$

Figure 6.11: Set of auxiliary definitions encoding the structural focusing rules in *LKF*. Here, the universal quantifiers around the definitions are elided; and $\bullet$ is either the connective & or $\parr$.

formulas. We use the judgment $\vdash_{lkf} [\Gamma], \Theta$ to denote that the sequent $\vdash [\Gamma], \Theta$ is provable in *LKF* and the judgment $\mapsto_{lkf} [\Gamma], \Theta$ to denote that the sequent $\mapsto [\Gamma], \Theta$ is provable in *LKF*. The structural rules D, R and [] enforce the following focusing discipline on *LKF* proofs: in all sequents, Θ is only composed of positive formulas and literals; a positive formula is introduced only when it is focused on and when there are only positive formulas or literals in the context; and negative non-literal formulas are introduced only when there is no formula focused on.

The theory $\mathcal{L}_{lkf}$ in Figure 6.10 encodes the system *LKF* by using two unrelated subexponentials: s which is unbounded and is used to encode the bracketed context of *LKF*'s sequents; and f which is bounded and is used to encode the stoup of *LKF*'s focused sequents. The structural rules D, R and [] are encoded implicitly by the use of the subexponentials s and f. The correct placement of these subexponentials depend on the type of the object-logic formulas involved in the clauses, which is specified by the set of definitions $str^a_{\&}$, $str^a_{\parr}$, str^s, and str^a. The definitions $str^a_{\&}$, $str^a_{\parr}$, and str^a are used in the encodings of asynchronous rules and specify that positive subformulas and literals are moved to the s-context. On the other hand, the definitions str^s are used in the encoding of synchronous rules and specify that positive subformulas and positive literals are moved to the f-context and that negative literals are moved to the s-context. Notice, however, that we use these definitions just to give a better presentation for the theory $\mathcal{L}_{lkf}$. One could alternatively present another theory, where these definitions are not necessary and that contains a clause for each possible combination of the principal formula's subformulas. For example, this theory would contain the following clause,

$$\exists A_1 A_2 B_1 B_2 [\lceil (A_1 \wedge^+ B_1) \wedge^+ (A_2 \vee^- B_2) \rceil^\perp \otimes (!^s ?^f \lceil A_1 \wedge^+ B_1 \rceil \otimes !^s \lceil A_2 \vee^- B_2 \rceil)],$$

which is obtained by replacing in the clause ($\wedge^+$) the atoms $str^s\ A_1 \wedge^+ B_1$ and $str^s\ A_2 \vee^- B_2$ by a simplification of the body of their definitions.

Strictly, we do not encode all *LKF* proofs but the *LKF* proofs that apply eagerly the structural focusing rules: $[]$, R, and D. One could construct a variant of *LKF* that incorporates these structural rules into its logical rules. For example, instead of the $\wedge^-$ rule in *LKF*, this system would contain different variants for this rule according to the type of subformulas of the introduced conjunction:

$$\frac{\vdash [\Theta, P_a^1], \Gamma \quad \vdash [\Theta], \Gamma, B_1}{\vdash [\Theta], \Gamma, P_a^1 \wedge^- B_1}\ [\wedge_1^-] \qquad \frac{\vdash [\Theta, P_a^1], \Gamma \quad \vdash [\Theta, P_a^2], \Gamma}{\vdash [\Theta], \Gamma, P_a^1 \wedge^- P_a^2}\ [\wedge_2^-]$$

$$\frac{\vdash [\Theta], \Gamma, B_1 \quad \vdash [\Theta, P_a^1], \Gamma}{\vdash [\Theta], \Gamma, B_1 \wedge^- P_a^1}\ [\wedge_3^-] \qquad \frac{\vdash [\Theta], \Gamma, B_1 \quad \vdash [\Theta], \Gamma, B_2}{\vdash [\Theta], \Gamma, B_1 \wedge^- B_2}\ [\wedge_4^-]$$

$$\frac{\vdash [\Theta, P_a^1] \quad \vdash [\Theta], B_1}{\mapsto [\Theta], P_a^1 \wedge^- B_1}\ [\wedge_5^-] \qquad \frac{\vdash [\Theta, P_a^1] \quad \vdash [\Theta, P_a^2]}{\mapsto [\Theta], P_a^1 \wedge^- P_a^2}\ [\wedge_6^-]$$

$$\frac{\vdash [\Theta], B_1 \quad \vdash [\Theta, P_a^1]}{\mapsto [\Theta], B_1 \wedge^- P_a^1}\ [\wedge_7^-] \qquad \frac{\vdash [\Theta], B_1 \quad \vdash [\Theta], B_2}{\mapsto [\Theta], B_1 \wedge^- B_2}\ [\wedge_8^-]$$

where P_a^1 and P_a^2 are positive or literal object-logic formulas and B_1 and B_2 are negative non-literal object-logic formulas. This explosion on the number of rules also occurred when Girard used his Unity of Logic system [Girard 1991] to define classical and intuitionistic connectives. As in the rules above, Girard used an introduction rule for each combination of polarities of the principal formula's subformulas.

To obtain an adequacy level of derivations, we assign negative polarity to all meta-level atoms, and since the structural focusing rules are encoded implicitly in the theory, we assume that the different contexts (bracketed, unbracketed, and stoup) contain the correct types of formulas.

Proposition 6.4 *Let all meta-level atoms be assigned negative polarity. Let Γ be a set of positive object-logic formulas or literals and Δ be a set of negative non-literal object-logic formulas and C be a positive object-logic formula. Let the subexponentials, s and f, be specified by the signature $\langle\{\infty, f, s\}; \{f \preceq \infty, s \preceq \infty\}\{s, \infty\}; \{s, \infty\}\rangle$. Then*

1. $\vdash_{lkf} [\Gamma], \Delta$ *iff* $\vdash_{sellf} \mathcal{L}_{lkf}\ \dot{s}\ \lceil\Gamma\rceil\ \dot{f}\ \cdot : \lceil\Delta\rceil \Uparrow$;
2. $\mapsto_{lkf} [\Gamma], C$ *iff* $\vdash_{sellf} \mathcal{L}_{lkf}\ \dot{s}\ \lceil\Gamma\rceil\ \dot{f}\ \lceil C\rceil : \cdot \Uparrow$.

Proof We prove both directions by structural induction on the height of proofs. The base case is when the derivation ends with an instance of the initial rule.

- Initial Rule:

$$\frac{}{\mapsto [\Theta, A^\perp], A}\ [\mathrm{I}] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{\dfrac{}{\vdash \mathcal{L}_{lkf}\ \dot{s}\ \lceil\Theta, lit^n\ A^\perp\rceil\ \dot{f}\ \lceil lit^p\ A\rceil : \cdot \Downarrow \lceil lit^p\ A\rceil^\perp}\ [I_f] \quad \dfrac{}{\vdash \mathcal{L}_{lkf}\ \dot{s}\ \lceil\Theta, lit^n\ A^\perp\rceil\ \dot{f}\ \cdot : \cdot \Downarrow \lceil lit^n\ A^\perp\rceil^\perp}\ [I_s]}{\vdash \mathcal{L}_{lkf}\ \dot{s}\ \lceil\Theta, lit^n\ A^\perp\rceil\ \dot{f}\ \lceil lit^p\ A\rceil : \cdot \Downarrow \lceil lit^p\ A\rceil^\perp \otimes \lceil lit^n\ A^\perp\rceil^\perp}\ [\otimes]}{\vdash \mathcal{L}_{lkf}\ \dot{s}\ \lceil\Theta, lit^n\ A^\perp\rceil\ \dot{f}\ \lceil lit^p\ A\rceil : \cdot \Uparrow}\ [D_\infty, \exists]$$

Similarly as in the encoding of *LJQ**, because the object-logic rule D is implicit in the theory, there is another derivation that introduces the formula Id_1, where the context for the subexponential f is empty. This derivation corresponds to the object-logic derivation composed of the rules D and I. The other base cases for the rules t_1 and $\perp$ are similar to the previous case.

We now proceed with the inductive cases, starting with the negative conjunction introduction rule.

• $\wedge^-$: Since the [] rules are implicit in the theory $\mathcal{L}_{lkf}$, there are four cases to consider obtained according to the type of subformulas, A and B, of the introduced conjunction. We consider the case when A is a negative non-literal object-logic formula and B is a positive object-logic formula. The other cases are similar.

$$\dfrac{\vdash [\Theta], \Gamma, A \quad \dfrac{\vdash [\Theta, B], \Gamma}{\vdash [\Theta], \Gamma, B}\,[R]}{\vdash [\Theta], \Gamma, A \wedge^- B}\,[\wedge^-] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{\dfrac{}{\vdash \Sigma \Downarrow \lceil A \wedge^- B \rceil^{\perp}}\,[\mathrm{I}] \qquad \dfrac{\dfrac{\dfrac{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta \rceil \stackrel{\cdot}{f} \cdot : \lceil \Gamma, A \rceil \Uparrow}{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta \rceil \stackrel{\cdot}{f} \cdot : \lceil \Gamma \rceil \Uparrow \lceil A \rceil}\,[R\Uparrow] \quad \dfrac{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta, B \rceil \stackrel{\cdot}{f} \cdot : \lceil \Gamma \rceil \Uparrow}{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta \rceil \stackrel{\cdot}{f} \cdot : \lceil \Gamma \rceil \Uparrow ?^s \lceil B \rceil}\,[?^s]}{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta \rceil \stackrel{\cdot}{f} \cdot : \lceil \Gamma \rceil \Downarrow \lceil A \rceil \,\&\, ?^s \lceil B \rceil}\,[R\Downarrow, \&]}{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta \rceil \stackrel{\cdot}{f} \cdot : \lceil \Gamma \rceil \Downarrow str^a_{\&}\, A\, B}\,[\otimes]}{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta \rceil \stackrel{\cdot}{f} \cdot : \lceil \Gamma, A \wedge^- B \rceil \Downarrow \lceil A \wedge^- B \rceil^{\perp} \otimes (str^a_{\&}\, A\, B)}\,[D_\infty, 2 \times \exists]}{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta \rceil \stackrel{\cdot}{f} \cdot : \lceil \Gamma, A \wedge^- B \rceil \Uparrow}$$

where Σ is the set of formulas $\mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta \rceil \stackrel{\cdot}{f} \cdot : \lceil A \wedge^- B \rceil$. Notice, however, that since the R rules are also implicit in the theory $\mathcal{L}_{lkf}$, the following derivation can also introduce the formula $(\wedge^-)$:

$$\dfrac{\dfrac{\dfrac{}{\vdash \Sigma \Downarrow \lceil A \wedge^- B \rceil^{\perp}}\,[\mathrm{I}] \qquad \dfrac{\dfrac{\dfrac{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta \rceil \stackrel{\cdot}{f} \lceil A \rceil \Uparrow}{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta \rceil \stackrel{\cdot}{f} \cdot : \Uparrow \lceil A \rceil}\,[R\Uparrow] \quad \dfrac{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta, B \rceil \stackrel{\cdot}{f} \cdot : \cdot \Uparrow}{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta \rceil \stackrel{\cdot}{f} \cdot : \cdot \Uparrow ?^s \lceil B \rceil}\,[?^s]}{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta \rceil \stackrel{\cdot}{f} \cdot : \cdot \Downarrow \lceil A \rceil \,\&\, ?^s \lceil B \rceil}\,[R\Downarrow, \&]}{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta \rceil \stackrel{\cdot}{f} \cdot : \cdot \Downarrow str^a_{\&}\, A\, B}\,[\otimes]}{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta \rceil \stackrel{\cdot}{f} \lceil A \wedge^- B \rceil : \cdot \Downarrow \lceil A \wedge^- B \rceil^{\perp} \otimes (str^a_{\&}\, A\, B)}\,[D_\infty, 2 \times \exists]}{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta \rceil \stackrel{\cdot}{f} \lceil A \wedge^- B \rceil : \cdot \Uparrow}$$

that corresponds to the object-logic derivation composed of the rules $R, \wedge^-$ and $[\,]$ that introduces the sequent $\mapsto [\Theta], A \wedge^- B$.

• $\vee^-$: Again there are four cases obtained according to the type the subformulas, A and B, of the disjunction introduced. We consider the case when A is a negative non-literal object-logic formula and B is a positive object-logic formula.

$$\dfrac{\dfrac{\vdash [\Theta, B], \Gamma, A}{\vdash [\Theta], \Gamma, A, B}\,[[]]}{\vdash [\Theta], \Gamma, A \vee^- B}\,[\vee^-] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{\dfrac{}{\vdash \Sigma : \lceil A \vee^- B \rceil \Downarrow \lceil A \vee^- B \rceil^{\perp}}\,[\mathrm{I}] \qquad \dfrac{\dfrac{\dfrac{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta, B \rceil \stackrel{\cdot}{f} \cdot : \lceil \Gamma, A \rceil \Uparrow}{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta \rceil \stackrel{\cdot}{f} \cdot : \lceil \Gamma \rceil \Uparrow \lceil A \rceil, ?^s \lceil B \rceil}\,[R\Uparrow, ?^s] }{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta \rceil \stackrel{\cdot}{f} \cdot : \lceil \Gamma \rceil \Downarrow \lceil A \rceil \parr ?^s \lceil B \rceil}\,[R\Downarrow, \parr]}{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta \rceil \stackrel{\cdot}{f} \cdot : \lceil \Gamma \rceil \Downarrow str^a_{\parr}\, A\, B}\,[\otimes]}{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta \rceil \stackrel{\cdot}{f} \cdot : \lceil \Gamma, A \vee^- B \rceil \Downarrow \lceil A \vee^- B \rceil^{\perp} \otimes (str^a_{\parr}\, A\, B)}\,[D_\infty, 2 \times \exists]}{\vdash \mathcal{L}_{lkf} \stackrel{\cdot}{s} \lceil \Theta \rceil \stackrel{\cdot}{f} \cdot : \lceil \Gamma, A \vee^- B \rceil \Uparrow}$$

where Σ is the set of formulas $\vdots^{s}\ \mathcal{L}_{lkf}, \lceil\Theta\rceil\ \vdots^{f}\ \cdot : \lceil A \vee^{-} B\rceil$. As with the case for negative conjunction introduction rule, since R rules are encoded implicitly in the theory $\mathcal{L}_{lkf}$, there is another derivation that introduces the formula $(\vee^{-})$, which corresponds to the object-logic derivation composed of the rules $R, \vee^{-}$ and $[]$, introducing the sequent $\mapsto [\Theta], A \vee^{-} B$.

The inductive cases for the other asynchronous rules follow a similar reasoning.

The cases for the synchronous rules are not different from the cases for the asynchronous rules shown before. We show here only the case for the positive conjunction introduction rule.

- $\wedge^{+}$: Assume below that A and B are both positive formulas.

$$\frac{\mapsto [\Theta], A \quad \mapsto [\Theta], B}{\mapsto [\Theta], A \wedge^{+} B}\ [\wedge^{+}] \qquad \rightsquigarrow$$

$$\dfrac{\dfrac{}{\vdash \Sigma \Downarrow \lceil A \wedge^{+} B\rceil^{\perp}}\ [I_f] \qquad \dfrac{\dfrac{\dfrac{\vdash \mathcal{L}_{lkf}\ \vdots^{s}\ \lceil\Theta\rceil\ \vdots^{f}\ \lceil A\rceil : \cdot \Uparrow}{\vdash \mathcal{L}_{lkf}\ \vdots^{s}\ \lceil\Theta\rceil\ \vdots^{f}\ \cdot : \cdot \Downarrow !^{s}?^{f}\lceil A\rceil}\ [!^{s}, ?^{f}]}{\vdash \mathcal{L}_{lkf}\ \vdots^{s}\ \lceil\Theta\rceil\ \vdots^{f}\ \cdot : \cdot \Downarrow str^{s}\ A} \qquad \dfrac{\dfrac{\vdash \mathcal{L}_{lkf}\ \vdots^{s}\ \lceil\Theta\rceil\ \vdots^{f}\ \lceil B\rceil : \cdot \Uparrow}{\vdash \mathcal{L}_{lkf}\ \vdots^{s}\ \lceil\Theta\rceil\ \vdots^{f}\ \cdot : \cdot \Downarrow !^{s}?^{f}\lceil B\rceil}\ [!^{s}, ?^{f}]}{\vdash \mathcal{L}_{lkf}\ \vdots^{s}\ \lceil\Theta\rceil\ \vdots^{f}\ \cdot : \cdot \Downarrow str^{s}\ B}}{\vdash \mathcal{L}_{lkf}\ \vdots^{s}\ \lceil\Theta\rceil\ \vdots^{f}\ \cdot : \cdot \Downarrow str^{s}\ A \otimes str^{s}\ B}\ [\otimes]}{\vdash \mathcal{L}_{lkf}\ \vdots^{s}\ \lceil\Theta\rceil\ \vdots^{f}\ \lceil A \wedge^{+} B\rceil : \cdot \Downarrow \lceil A \wedge^{+} B\rceil^{\perp} \otimes (str^{s}\ A \otimes str^{s}\ B)}\ [\otimes]}{\vdash \mathcal{L}_{lkf}\ \vdots^{s}\ \lceil\Theta\rceil\ \vdots^{f}\ \lceil A \wedge^{+} B\rceil : \cdot \Uparrow}\ [D_{\infty}, 2 \times \exists]$$

where Σ is the set of formulas $\mathcal{L}_{lkf}\ \vdots^{s}\ \lceil\Theta\rceil\ \vdots^{f}\ \lceil A \wedge^{+} B\rceil$. Here the s-bang in the formula $(\wedge^{+})$ ensures that there is at most one formula in the context for the subexponential f, which corresponds in the object-logic that there is at most one formula focused on and that the unbracketed context is empty. As before, since the D rules are encoded implicitly in the theory $\mathcal{L}_{lkf}$, there is another derivation that introduces the formula $(\wedge^{+})$, which corresponds to the object-logic derivation composed of the rules D and $\wedge^{+}$. □

6.7 LJF

The focused system *LJF* for intuitionistic logic, also proposed by Liang & Miller [Liang 2008], is depicted in Figure 6.12. We assign arbitrarily the polarity for atoms and classify as positive the formulas whose main connective is either $\wedge^{+}$, $\vee$, $\exists$ or t and the positive polarity atoms. The remaining formulas are classified as negative. As before, this classification is specified by the first set of definitions shown in 6.14. Moreover, we encode atoms by using the terms $atom^{p}$ and $atom^{n}$, whose superscripts denote the polarities of the object-level atom: p for positive polarity atom and n for negative polarity atoms. Differently from *LKF*, the system *LJF* has two sided sequents, with the restriction that their right-hand-side contexts have at most one formula, and four types of sequents, described in Section 3.4.

We show that the theory $\mathcal{L}_{ljf}$ in Figure 6.7 encodes *LJF*. It uses three subexponentials: l is an unbounded subexponential which, intuitively, encodes the left bracketed context; r is a bounded subexponential, which encodes the right bracketed context; and f is a bounded subexponential that encodes the focusing stoup. As before with the encoding of *LKF*, the structural rules $[]_l, []_r, R_l, R_r, D_l$ and D_r in *LJF* are implicitly encoded in the theory $\mathcal{L}_{ljf}$, by using the defined atoms str^{sl}, str^{sr}, str^{al}, $str^{al}_{\&}$, $str^{al}_{\otimes}$, $str^{ar}_{\&}$, $str^{ar}_{\otimes}$, and str^{a}_{r} to correctly place subexponentials connectives. The definitions str^{sl} and str^{sr} are used in the encoding of introduction rules appearing in the synchronous phase and specify that formulas are moved to the f-context and atoms are moved to the l-context or r-context. On the other hand, the definitions str^{al}, $str^{al}_{\&}$, $str^{al}_{\otimes}$, $str^{ar}_{\&}$, $str^{ar}_{\otimes}$, and str^{a}_{r} are used in the encoding of introduction rules appearing in the asynchronous phase and specify that formulas and atoms are moved to the l-context or r-context. As in the encoding of *LKF*, we do not strictly encode *LJF* but a variant of it that incorporates the structural focusing rules into its logical rules.

$$\frac{[N,\Gamma] \xrightarrow{N} [R]}{[N,\Gamma] \longrightarrow [R]}\ [D_l] \qquad \frac{[\Gamma] -_P\!\!\rightarrow}{[\Gamma] \longrightarrow [P]}\ [D_r] \qquad \frac{[\Gamma], P \longrightarrow [R]}{[\Gamma] \xrightarrow{P} [R]}\ [R_l] \qquad \frac{[\Gamma] \longrightarrow N}{[\Gamma] -_N\!\!\rightarrow}\ [R_r]$$

$$\frac{}{[\Gamma] \xrightarrow{A_n} [A_n]}\ [I_l] \qquad \frac{}{[\Gamma, A_p] -_{A_p}\!\!\rightarrow}\ [I_r] \qquad \frac{[\Gamma, N_a], \Theta \longrightarrow \mathcal{R}}{[\Gamma], \Theta, N_a \longrightarrow \mathcal{R}}\ [[]_l] \qquad \frac{[\Gamma], \Theta \longrightarrow [P_a]}{[\Gamma], \Theta \longrightarrow P_a}\ [[]_r]$$

$$\frac{}{[\Gamma], \Theta, \bot \longrightarrow \mathcal{R}}\ [\bot_l] \qquad \frac{[\Gamma], \Theta \longrightarrow \mathcal{R}}{[\Gamma], \Theta, t \longrightarrow \mathcal{R}}\ [t_l] \qquad \frac{}{[\Gamma] -_t\!\!\rightarrow}\ [t_r]$$

$$\frac{[\Gamma], \Theta, A, B \longrightarrow \mathcal{R}}{[\Gamma], \Theta, A \wedge^+ B \longrightarrow \mathcal{R}}\ [\wedge_l^+] \qquad \frac{[\Gamma] -_A\!\!\rightarrow \quad [\Gamma] -_B\!\!\rightarrow}{[\Gamma] -_{A \wedge^+ B}\!\!\rightarrow}\ [\wedge_r^+] \qquad \frac{[\Gamma] -_A\!\!\rightarrow \quad [\Gamma] \xrightarrow{B} [R]}{[\Gamma] \xrightarrow{A \supset B} [R]}\ [\supset_l]$$

$$\frac{[\Gamma] \xrightarrow{A_i} R}{[\Gamma] \xrightarrow{A_1 \wedge^- A_2} [R]}\ [\wedge_l^-] \qquad \frac{[\Gamma], \Theta \longrightarrow A \quad [\Gamma], \Theta \longrightarrow B}{[\Gamma], \Theta \longrightarrow A \wedge^- B}\ [\wedge_r^-] \qquad \frac{[\Gamma], \Theta, A \longrightarrow B}{[\Gamma], \Theta \longrightarrow A \supset B}\ [\supset_r]$$

$$\frac{[\Gamma], \Theta, A \longrightarrow \mathcal{R} \quad [\Gamma], \Theta, B \longrightarrow \mathcal{R}}{[\Gamma], \Theta, A \vee B \longrightarrow \mathcal{R}}\ [\vee_l] \qquad \frac{[\Gamma] -_{A_i}\!\!\rightarrow}{[\Gamma] -_{A_1 \vee A_2}\!\!\rightarrow}\ [\vee_i] \qquad \frac{[\Gamma] \xrightarrow{A\{t/x\}} [R]}{[\Gamma] \xrightarrow{\forall x A} [R]}\ [\forall_l]$$

$$\frac{[\Gamma], \Theta, A \longrightarrow \mathcal{R}}{[\Gamma], \Theta, \exists x A\{c/x\} \longrightarrow \mathcal{R}}\ [\exists_l] \qquad \frac{[\Gamma] -_{A[t/x]}\!\!\rightarrow}{[\Gamma] -_{\exists x A}\!\!\rightarrow}\ [\exists_r] \qquad \frac{[\Gamma], \Theta \longrightarrow A\{c/x\}}{[\Gamma], \Theta \longrightarrow \forall y A}\ [\forall_r]$$

Figure 6.12: *LJF*: Here, Γ and Θ are multisets of formulas, A_n denotes a negative atom, A_p a positive atom, and P a positive formula, N a negative formula, N_a a negative formula or an atom, and P_a a positive formula or an atom. All other formulas are arbitrary and y is not free in Γ, Θ or R.

To obtain an adequacy level of derivations, we assume that the different contexts (bracketed, unbracketed, and stoup) contain the correct types of formulas and assign negative polarity to all meta-level atoms, as states the following proposition.

Proposition 6.5 *Let all meta-level atoms, $\lceil\cdot\rceil$ and $\lfloor\cdot\rfloor$, be assigned negative polarity. Let Γ be a set of negative object-logic formulas or atoms, Δ be a set of positive non-atomic object-logic formulas, C_p be a positive object-logic formula, C_n be a negative object-logic formula, P be positive object-logic formula or an atom, and N be a negative non-atomic object-logic formula. Let the subexponentials, l, r and f, be specified by the signature $\langle\{f, r, l, \infty\}\{f \preceq \infty, r \prec l \preceq \infty\}; \{l, \infty\}; \{l, \infty\}\rangle$. Then*

1. $[\Gamma], \Delta \longrightarrow N$ *is provable in LJF iff* $\vdash_{\mathsf{sellf}} \mathcal{L}_{ljf}\ \vdots_l\ \lfloor\Gamma\rfloor\ \vdots_r\ \cdot\ \vdots_f\ \cdot : \lfloor\Delta\rfloor, \lceil N\rceil \Uparrow$;
2. $[\Gamma], \Delta \longrightarrow [P]$ *is provable in LJF iff* $\vdash_{\mathsf{sellf}} \mathcal{L}_{ljf}\ \vdots_l\ \lfloor\Gamma\rfloor\ \vdots_r\ \lceil P\rceil\ \vdots_f\ \cdot : \lfloor\Delta\rfloor \Uparrow$;
3. $[\Gamma] -_{C_p}\!\!\rightarrow$ *is provable in LJF iff* $\vdash_{\mathsf{sellf}} \mathcal{L}_{ljf}\ \vdots_l\ \lfloor\Gamma\rfloor\ \vdots_r\ \cdot\ \vdots_f\ \lceil C_p\rceil : \cdot \Uparrow$;
4. $[\Gamma] \xrightarrow{C_n} [P]$ *is provable in LJF iff* $\vdash_{\mathsf{sellf}} \mathcal{L}_{ljf}\ \vdots_l\ \lfloor\Gamma\rfloor\ \vdots_r\ \lceil P\rceil\ \vdots_f\ \lfloor C_n\rfloor : \cdot \Uparrow$.

Proof Again the proof follows by structural induction on the height of proofs. The proof follows the same reasoning as in the proof of Proposition 6.4. In all the inductive cases there are several cases to be considered, obtained according to the type of subformulas of the principal formula. Here we show only some cases, starting with the inductive case for the left implication introduction rule.

- $\supset_l$: Here we assume that A is a positive formula and B is a negative formula. The other cases are similar.

$(\wedge_l^+)$	$\lfloor A \wedge^+ B \rfloor^\perp \otimes (str^{al}_{\parr}\, A\, B)$	$(\wedge_r^+)$	$\lceil A \wedge^+ B \rceil^\perp \otimes (str^{sr}\, A \otimes str^{sr}\, B)$
$(\wedge_l^-)$	$\lfloor A_1 \wedge^- A_2 \rfloor^\perp \otimes (str^{sl}\, A_1 \oplus str^{sl}\, A_2)$	$(\wedge_r^-)$	$\lceil A \wedge^- B \rceil^\perp \otimes (str^{ar}_{\&}\, A\, B)$
$(\vee_l)$	$\lfloor A \vee B \rfloor^\perp \otimes (str^{al}_{\&}\, A\, B)$	$(\vee_r)$	$\lceil A_1 \vee A_2 \rceil^\perp \otimes (str^{sr}\, A_1 \oplus str^{sr}\, A_2)$
$(\supset_l)$	$\lfloor A \supset B \rfloor^\perp \otimes (str^{sr}\, A \otimes str^{sl}\, B)$	$(\supset_r)$	$\lceil A \supset B \rceil^\perp \otimes (str^{ar}_{\parr}\, A\, B)$
$(\exists_l)$	$\lfloor \exists B \rfloor^\perp \otimes (str^{al}\, B)$	$(\exists_r)$	$\lceil \exists B \rceil^\perp \otimes (str^{sr}\, Bx)$
$(\forall_l)$	$\lfloor \forall B \rfloor^\perp \otimes (str^{sl}\, Bx)$	$(\forall_r)$	$\lceil \forall B \rceil^\perp \otimes (str^{a}_{r}\, B)$
(t_l)	$\lfloor t \rfloor^\perp \otimes \perp$	(t_r)	$\lceil t \rceil^\perp \otimes !_s \top$
$(\perp_l)$	$\lfloor \perp \rfloor \otimes \top$		
(Id_1^p)	$\lceil atom^p\, A \rceil^\perp \otimes \lfloor atom^p\, A \rfloor^\perp$	(Id_1^n)	$\lceil atom^n\, A \rceil^\perp \otimes \lfloor atom^n\, A \rfloor^\perp$

Figure 6.13: The theory $\mathcal{L}_{ljf}$ used to encode *LJF*.

$$
\begin{array}{lcl}
syn\ A & \triangleq & \exists A_1 A_2 (A = A_1 \wedge^+ A_2) \oplus \exists A_1 A_2 (A = A_1 \vee A_2) \oplus \exists B (A = \exists B) \oplus (A = t) \\
asyn\ A & \triangleq & \exists A_1 A_2 (A = A_1 \wedge^- A_2) \oplus \exists A_1 A_2 (A = A_1 \supset A_2) \oplus \exists B (A = \forall B) \oplus (A = \perp) \\
atom\ A & \triangleq & \exists A_1 [(A = atom^n\, A_1) \oplus (A = atom^p\, A_1)] \\
atSyn\ A & \triangleq & \exists A_1 [(atom\ A_1) \oplus (syn\ A_1)] \\
atAsyn\ A & \triangleq & \exists A_1 [(atom\ A_1) \oplus (asyn\ A_1)] \\
pos\ A & \triangleq & \exists A_1 [(A = atom^p\, A_1) \oplus (syn\ A_1)] \\
neg\ A & \triangleq & \exists A_1 [(A = atom^n\, A_1) \oplus (asyn\ A_1)]
\end{array}
$$

$$
\begin{array}{lcl}
str^{al}_{\bullet}\, A\, B & \triangleq & [(atAsyn\ A) \otimes (atAsyn\ B) \otimes (?^l \lfloor A \rfloor \bullet ?^l \lfloor B \rfloor)] \oplus \\
& & [(atAsyn\ A) \otimes (syn\ B) \otimes (?^l \lfloor A \rfloor \bullet \lfloor B \rfloor)] \oplus \\
& & [(syn\ A) \otimes (atAsyn\ B) \otimes (\lfloor A \rfloor \bullet ?^l \lfloor B \rfloor)] \oplus \\
& & [(syn\ A) \otimes (syn\ B) \otimes (\lfloor A \rfloor \bullet \lfloor B \rfloor)]
\end{array}
$$

$$
\begin{array}{lcl}
str^{al}\, A & \triangleq & \exists B (A = \lambda x B) \otimes \{[atAsyn\ B \otimes (\forall x ?^l \lfloor Ax \rfloor)] \oplus [syn\ B \otimes (\forall x \lfloor Ax \rfloor)]\}
\end{array}
$$

$$
\begin{array}{lcl}
str^{sl}\, A & \triangleq & [(neg\ A) \otimes (!^r\, ?^f \lfloor A \rfloor)] \oplus [(syn\ A) \otimes (!^r \lfloor A \rfloor)] \oplus [\exists A_1 (A = atom^p\, A_1) \otimes (!^r ?^l \lfloor A \rfloor)]
\end{array}
$$

$$
\begin{array}{lcl}
str^{ar}_{\&}\, A\, B & \triangleq & [(atSyn\ A) \otimes (atSyn\ B) \otimes (?^r \lceil A \rceil\ \&\ ?^r \lceil B \rceil)] \oplus \\
& & [(atSyn\ A) \otimes (asyn\ B) \otimes (?^r \lceil A \rceil\ \&\ \lceil B \rceil)] \oplus \\
& & [(asyn\ A) \otimes (atSyn\ B) \otimes (\lceil A \rceil\ \&\ ?^r \lceil B \rceil)] \oplus \\
& & [(asyn\ A) \otimes (asyn\ B) \otimes (\lceil A \rceil\ \&\ \lceil B \rceil)]
\end{array}
$$

$$
\begin{array}{lcl}
str^{ar}_{\parr}\, A\, B & \triangleq & [(atAsyn\ A) \otimes (atSyn\ B) \otimes (?^l \lfloor A \rfloor\ \parr\ ?^r \lceil B \rceil)] \oplus \\
& & [(atAsyn\ A) \otimes (asyn\ B) \otimes (?^l \lfloor A \rfloor\ \parr\ \lceil B \rceil)] \oplus \\
& & [(syn\ A) \otimes (atSyn\ B) \otimes (\lfloor A \rfloor\ \parr\ ?^r \lceil B \rceil)] \oplus \\
& & [(syn\ A) \otimes (asyn\ B) \otimes (\lfloor A \rfloor\ \parr\ \lceil B \rceil)]
\end{array}
$$

$$
\begin{array}{lcl}
str^{a}_{r}\, A & \triangleq & \exists B (A = \lambda x B) \otimes \{[atSyn\ B \otimes (\forall x ?^r \lceil Ax \rceil)] \oplus [asyn\ B \otimes (\forall x \lceil Ax \rceil)]\}
\end{array}
$$

$$
\begin{array}{lcl}
str^{sr}\, A & \triangleq & [(pos\ A) \otimes (!^l\, ?^f \lceil A \rceil)] \oplus [(asyn\ A) \otimes (!^l \lceil A \rceil)] \oplus [\exists A_1 (A = atom^n\, A_1) \otimes (!^l ?^r \lceil A \rceil)]
\end{array}
$$

Figure 6.14: Set of auxiliary definitions encoding the structural focusing rules in *LJF*. Here, the universal quantifiers around the definitions are elided; and $\bullet$ is either the connective & or $\parr$.

$$\frac{[\Gamma]-_A\rightarrow \quad [\Gamma] \xrightarrow{B} [R]}{[\Gamma] \xrightarrow{A\supset B} [R]}\,[\supset_l] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \Sigma \Downarrow \lfloor A \supset B\rfloor^{\perp}}\,[I_f] \quad \dfrac{\dfrac{\dfrac{\vdash \mathcal{L}_{ljf}\ \dot{l}\ \lfloor\Gamma\rfloor\ \dot{r}\cdot\dot{f}\ \lceil A\rceil : \cdot \Uparrow}{\vdash \mathcal{L}_{ljf}\ \dot{l}\ \lfloor\Gamma\rfloor\ \dot{r}\cdot\dot{f}\cdot : \cdot \Uparrow ?^f\lceil A\rceil}\,[?^f]}{\vdash \mathcal{L}_{ljf}\ \dot{l}\ \lfloor\Gamma\rfloor\ \dot{r}\cdot\dot{f}\cdot : \cdot \Downarrow !^l?^f\lceil A\rceil}\,[!^l]}{\vdash \mathcal{L}_{ljf}\ \dot{l}\ \lfloor\Gamma\rfloor\ \dot{r}\cdot\dot{f}\cdot : \cdot \Downarrow str^{sr}\ A} \quad \dfrac{\dfrac{\dfrac{\vdash \mathcal{L}_{ljf}\ \dot{l}\ \lfloor\Gamma\rfloor\ \dot{r}\ \lceil R\rceil\ \dot{f}\ \lfloor B\rfloor : \cdot \Uparrow}{\vdash \mathcal{L}_{ljf}\ \dot{l}\ \lfloor\Gamma\rfloor\ \dot{r}\ \lceil R\rceil\ \dot{f}\cdot : \cdot \Uparrow ?^f\lfloor B\rfloor}\,[?^f]}{\vdash \mathcal{L}_{ljf}\ \dot{l}\ \lfloor\Gamma\rfloor\ \dot{r}\ \lceil R\rceil\ \dot{f}\cdot : \cdot \Downarrow !^r?^f\lfloor B\rfloor}\,[!^r]}{\vdash \mathcal{L}_{ljf}\ \dot{l}\ \lfloor\Gamma\rfloor\ \dot{r}\ \lceil R\rceil\ \dot{f}\cdot : \cdot \Downarrow str^{sl}\ B}}{\vdash \mathcal{L}_{ljf}\ \dot{l}\ \lfloor\Gamma\rfloor\ \dot{r}\ \lceil R\rceil\ \dot{f}\cdot : \cdot \Downarrow str^{sr}\ A \otimes str^{sl}\ B}\,[\otimes]}{\vdash \mathcal{L}_{ljf}\ \dot{l}\ \lfloor\Gamma\rfloor\ \dot{r}\ \lceil R\rceil\ \dot{f}\ \lfloor A\supset B\rfloor : \cdot \Downarrow \lceil A\supset B\rceil^{\perp} \otimes (str^{sr}\ A \otimes str^{sl}\ B)}\,[\otimes]}{\vdash \mathcal{L}_{ljf}\ \dot{l}\ \lfloor\Gamma\rfloor\ \dot{r}\ \lceil R\rceil\ \dot{f}\ \lfloor A\supset B\rfloor : \cdot \Uparrow}\,[D_\infty, 2\times\exists]$$

where Σ is the set of formulas $\mathcal{L}_{ljf}\ \dot{l}\ \lfloor\Gamma\rfloor\ \dot{r}\cdot\dot{f}\ \lfloor A\supset B\rfloor$. Here, the role of the subexponential l and r-bangs is most important: the latter bang ensures that the f-context in the right-most branch is empty, and the former bang ensures that both the r and f contexts, in the middle branch, are empty, enforcing that the atom $\lfloor A\supset B\rfloor$ is consumed in the left-most branch and that the atom $\lceil R\rceil$ is moved to the right-most branch. As the D_l rules are implicit in the theory $\mathcal{L}_{ljf}$, there is another derivation that also introduces the formula $(\supset_l)$, which corresponds to the object-logic derivation composed of the rules D_l and $\supset_l$.

- $\vee_1$: Assume here that A_1 is a positive polarity formula. The other cases are similar.

$$\frac{[\Gamma]-_{A_1}\rightarrow}{[\Gamma]-_{A_1\vee A_2}\rightarrow}\,[\vee_1] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \Sigma \Downarrow \lceil A_1\vee A_2\rceil^{\perp}}\,[I_f] \quad \dfrac{\dfrac{\dfrac{\vdash \mathcal{L}_{ljf}\ \dot{l}\ \lfloor\Gamma\rfloor\ \dot{r}\cdot\dot{f}\ \lceil A_1\rceil : \cdot \Uparrow}{\vdash \mathcal{L}_{ljf}\ \dot{l}\ \lfloor\Gamma\rfloor\ \dot{r}\cdot\dot{f}\cdot : \cdot \Uparrow ?^f\lceil A_1\rceil}\,[?^f]}{\vdash \mathcal{L}_{ljf}\ \dot{l}\ \lfloor\Gamma\rfloor\ \dot{r}\cdot\dot{f}\cdot : \cdot \Downarrow !^l?^f\lceil A_1\rceil}\,[!^l]}{\vdash \mathcal{L}_{ljf}\ \dot{l}\ \lfloor\Gamma\rfloor\ \dot{r}\cdot\dot{f}\cdot : \cdot \Downarrow str^{sr}\ A_1}}{\vdash \mathcal{L}_{ljf}\ \dot{l}\ \lfloor\Gamma\rfloor\ \dot{r}\cdot\dot{f}\ \lceil A_1\vee A_2\rceil : \cdot \Downarrow \lceil A_1\vee A_2\rceil^{\perp} \otimes (str^{sr}\ A_1 \oplus str^{sr}\ A_2)}\,[\otimes]}{\vdash \mathcal{L}_{ljf}\ \dot{l}\ \lfloor\Gamma\rfloor\ \dot{r}\cdot\dot{f}\ \lceil A_1\vee A_2\rceil : \cdot \Uparrow}\,[D_\infty, 2\times\exists]$$

where Σ is the set of formulas $\mathcal{L}_{ljf}\ \dot{l}\ \lfloor\Gamma\rfloor\ \dot{r}\cdot\dot{f}\ \lceil A_1\vee A_2\rceil$. Once again the role of the subexponentials is crucial for the correct encoding of the object-logic rule: the l-bang ensures that the subexponential contexts for f and r are empty. There are other derivations that introduce the formula $(\vee_r)$, which correspond to the object-logic derivations composed of D and $\vee_{ri}$, for $i\in\{1,2\}$. These derivations and the remaining rules appearing in the synchronous phase are similar to the two cases above.

- $\wedge_l^+$: Assume below that A is a positive non-atomic formula and B is a negative formula.

$$\dfrac{\dfrac{[\Gamma, B], \Theta, A \longrightarrow R}{[\Gamma], \Theta, A, B \longrightarrow R}\,[[]_l]}{[\Gamma], \Theta, A \wedge^+ B \longrightarrow R}\,[\wedge_l^+] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{\vdash \Sigma \Downarrow \lfloor A \wedge^{+} B \rfloor^{\perp}}{\vdash \Sigma \Downarrow \lfloor A \wedge^{+} B \rfloor^{\perp}}\,[\mathrm{I}] \quad \dfrac{\dfrac{\dfrac{\vdash \mathcal{L}_{ljf}\ \dot{i}\ \lfloor \Gamma, B \rfloor\ \dot{r} \cdot \dot{f} \cdot : \lfloor \Theta, A \rfloor, \lceil R \rceil \Uparrow}{\vdash \mathcal{L}_{ljf}\ \dot{i}\ \lfloor \Gamma \rfloor\ \dot{r} \cdot \dot{f} \cdot : \lfloor \Theta \rfloor, \lceil R \rceil \Uparrow \lfloor A \rfloor, ?^{l} \lfloor B \rfloor}\,[R\Uparrow, ?^{l}]}{\vdash \mathcal{L}_{ljf}\ \dot{i}\ \lfloor \Gamma \rfloor\ \dot{r} \cdot \dot{f} \cdot : \lfloor \Theta \rfloor, \lceil R \rceil \Downarrow \lfloor A \rfloor \parr ?^{l} \lfloor B \rfloor}\,[R\Downarrow, \parr]}{\vdash \mathcal{L}_{ljf}\ \dot{i}\ \lfloor \Gamma \rfloor\ \dot{r} \cdot \dot{f} \cdot : \lfloor \Theta \rfloor, \lceil R \rceil \Downarrow str^{al}_{\&}\, A\, B}\,[\otimes]}{\vdash \mathcal{L}_{ljf}\ \dot{i}\ \lfloor \Gamma \rfloor\ \dot{r} \cdot \dot{f} \cdot : \lfloor \Theta, A \wedge^{+} B \rfloor, \lceil R \rceil \Downarrow \lfloor A \wedge^{+} B \rfloor^{\perp} \otimes (str^{al}_{\&}\, A\, B)}\,[D_{\infty}, 2 \times \exists]}{\vdash \mathcal{L}_{ljf}\ \dot{i}\ \lfloor \Gamma \rfloor\ \dot{r} \cdot \dot{f} \cdot : \lfloor \Theta, A \wedge^{+} B \rfloor, \lceil R \rceil \Uparrow}$$

where Σ is the set of formulas $\dot{i}\ \mathcal{L}_{ljf}, \lfloor \Gamma \rfloor\ \dot{r} \cdot \dot{f} \cdot : \lfloor A \wedge^{+} B \rfloor$. Since the rules R_l are implicit in the theory $\mathcal{L}_{ljf}$, there is another derivation that introduces the formula $(\wedge^{+}_{l})$, which corresponds to the object-logic derivation composed of the rules R_l and $\wedge^{+}_{l}$.

- $\wedge^{-}_{r}$: Assume that both A and B are negative non-atomic formulas.

$$\dfrac{[\Gamma], \Theta \longrightarrow A \quad [\Gamma], \Theta \longrightarrow B}{[\Gamma], \Theta \longrightarrow A \wedge^{-} B}\,[\wedge^{-}_{r}] \qquad \leftrightsquigarrow$$

$$\dfrac{\dfrac{\vdash \mathcal{K} : \lfloor \Theta \rfloor, \lceil A \rceil \Uparrow}{\vdash \mathcal{K} : \lfloor \Theta \rfloor \Uparrow \lceil A \rceil}\,[R\Uparrow] \quad \dfrac{\vdash \mathcal{K} : \lfloor \Theta \rfloor, \lceil B \rceil \Uparrow}{\vdash \mathcal{K} : \lfloor \Theta \rfloor \Uparrow \lceil B \rceil}\,[R\Uparrow]}{\vdash \mathcal{L}_{ljf}\ \dot{i}\ \lfloor \Gamma \rfloor\ \dot{r} \cdot \dot{f} \cdot : \lfloor \Theta \rfloor \Downarrow \lceil A \rceil \,\&\, \lceil B \rceil}\,[R\Downarrow, \&]$$

$$\dfrac{\dfrac{\vdash \mathcal{K} : \lceil A \wedge^{-} B \rceil \Downarrow \lceil A \wedge^{-} B \rceil^{\perp}}{\vdash \mathcal{K} : \lceil A \wedge^{-} B \rceil \Downarrow \lceil A \wedge^{-} B \rceil^{\perp}}\,[\mathrm{I}] \quad \dfrac{\vdash \mathcal{L}_{ljf}\ \dot{i}\ \lfloor \Gamma \rfloor\ \dot{r} \cdot \dot{f} \cdot : \lfloor \Theta \rfloor \Downarrow str^{ar}_{\&}\, A\, B}{\vdash \mathcal{L}_{ljf}\ \dot{i}\ \lfloor \Gamma \rfloor\ \dot{r} \cdot \dot{f} \cdot : \lfloor \Theta \rfloor, \lceil A \wedge^{-} B \rceil \Downarrow \lceil A \wedge^{-} B \rceil^{\perp} \otimes (str^{ar}_{\&}\, A\, B)}\,[\otimes]}{\vdash \mathcal{L}_{ljf}\ \dot{i}\ \lfloor \Gamma \rfloor\ \dot{r} \cdot \dot{f} \cdot : \lfloor \Theta \rfloor, \lceil A \wedge^{-} B \rceil \Uparrow}\,[D_{\infty}, 2 \times \exists]$$

where $\mathcal{K}$ is the set of formulas $\dot{i}\ \mathcal{L}_{ljf}, \lfloor \Gamma \rfloor\ \dot{r} \cdot \dot{f} \cdot$. Since the rules R_l are implicit in the theory $\mathcal{L}_{ljf}$, there is another derivation that introduces the formula $(\wedge^{-}_{r})$, which corresponds to the object-logic derivation composed of the rules R_l and $\wedge^{-}_{r}$. □

6.8 Other Focused Proof Systems

Liang & Miller also show in [Liang 2008, Liang 2007] how to encode the focused systems *LJQ'* [Dyckhoff 2006], *LJT* [Herbelin 1995] and λRCC [Jagadeesan 2005] in *LJF*. They encoded these systems by modifying the synthetic connective of object-logic formulas, by, for example, changing the polarity assignment of atoms, or by using equivalent formulas that might have different focusing behaviors, such as the formulas F and $F \wedge^{+} t$, when F is a negative formula. Given the encoding $\mathcal{L}_{ljf}$ for *LJF*, one can easily obtain encodings for these other focused proof systems by using the same translations used by Liang & Miller.

6.9 Conclusions and future works

We have illustrated that, by using subexponentials, a wide range of proofs systems can be encoded in the same declarative way as done in Chapter 4 and with the strongest level of adequacy – the full completeness of derivations. For example, we encoded the *G1* sequent calculus for minimal, intuitionistic and classical logic; the multi-conclusion system for intuitionistic logic, *mLJ*; and several focused proof systems: *LJQ**, *LJF* and *LKF*. The subexponentials present in the theories encoding these logics played an important role to correctly specify the structural restrictions of these systems,

such as, in intuitionistic logic, the right-hand-side of sequents contain at most one formula or, in the focused systems, the decide, release and react rules that specify the focusing discipline on proofs.

Although these theories are declarative and describe exactly the proofs systems encoded, we have not yet given equivalence results between any two theories of the same logic, that is, for example, the theory for *LKF* proves the same theorems as the theory for *G1c*. One consequence of such equivalences would correspond to the focusing theorem for *LKF* stating the completeness of the focusing discipline. However, the equivalence arguments seem to be harder than the reasoning used in Chapter 4, as one would have to relate specifications that use different subexponentials. As future work one could try to relate different theories by using linear logic equivalences or simple proofs by induction.

Miller & Pimentel [Miller 2002] presented necessary conditions for assuring that an encoded system satisfies cut-elimination. In that paper, they assumed linear logic theories that did not contain subexponentials. A direction for future work could be to use the dualities in linear logic with subexponentials to find similar conditions for systems encoded in linear logic with subexponentials.

Many other systems have structural restrictions that seem hard to specify by using subexponentials. For example, in the *G3* systems [Troelstra 1996], the main formula of an inference rule is consumed, while the side formulas are contracted. Subexponentials do not seem to capture this type of behavior because contexts can only be seen either as sets or multisets, and therefore either the whole context is contracted, including the encoding of the main formula, or no formula in the context is contracted.

Miller & Saurin proposed, in [Miller 2007b], the multifocusing system for linear logic that allows the focusing on more than one formula. We do not encode this system here, but it seems to be possible to do so, by using, instead of *SELLF*, its multifocused version. One still would need to understand how to relate a synchronous derivations in the meta-logic with object-logic derivations.

Chapter 7

Algorithmic specifications in linear logic with subexponentials

In this chapter, we continue to exploit the increase of expressiveness obtained by using linear logic with subexponentials for the specification of algorithms. In particular, we use *subexponentials* to assign *locations* to multisets of formulas within a linear logic programming setting. Treating locations as subexponentials greatly increases the algorithmic expressiveness of logic. To illustrate this new expressiveness, we show that focused proof search can be precisely linked to a simple algorithmic specification language that contains while-loops, conditionals, and insertion into and deletion from multisets. We also give some general conditions for when a focused proof step can be executed in constant time.

Reference: parts of this chapter appeared in [Nigam 2009a].

7.1 Introduction

Computation in the *proof-search* paradigm (a.k.a. logic programming) can be characterized as the process of searching for a cut-free sequent proof. The expressiveness of logic programming can be judged, in part, by examining the kind of changes that can take place within sequents during the search for a proof. Let Ξ be a cut-free proof of $\Gamma \vdash \Delta$ and let $\Gamma' \vdash \Delta'$ be a sequent occurring in Ξ. The dynamics of proof search in this setting can be partially judged by examining the possible differences that can occur between Γ and Γ' and between Δ and Δ'.

When proof search is conducted within intuitionistic logic, Γ is usually treated as a set of formulas and Δ as a single formula. If we restrict further to Horn clauses, we find that $\Gamma = \Gamma'$ and that Δ and Δ' are atomic formulas. Thus, the only real dynamics during proof search with Horn clauses is that atomic goals change as we move upward through a proof. As a result, all data structures and their various relationships must be encoded as terms within atomic formulas: that is, all the dynamics of computation is buried within *non-logical contexts* (within the scope of predicates). If one uses hereditary Harrop formulas instead of Horn clauses, slightly richer changes are possible: in particular, $\Gamma \subseteq \Gamma'$. When proof search is conducted within linear logic, both Γ and Δ can be treated as multisets and the logic program is free to specify arbitrary, computable relationships between Γ and Γ' and between Δ and Δ'. In linear logic, some data structures and their relationships can be encoded directly in the *logical context* of proofs.

Of course, many data structures can be encoded naturally as sets or multisets of atomic formulas: for example, a graph given by a set of nodes N and an adjacency relation A can be encoded as the multiset of atomic formulas

$$\{node\ x \mid x \in N\} \cup \{adj\ x\ y \mid \langle x, y\rangle \in A\},$$

where *node* and *adj* are predicates. A major obstacle to describing algorithms using linear logic programs is that data encoded into contexts does not support enough tests on data. While it is possible in linear logic to detect that the global multiset context is empty, it is not possible to

perform this test on less than the entire context. Given the multiset encoding of graphs above, linear logic provides a simple mechanism to detect that both the set of nodes and the adjacency information are empty but the logic does not provide a means to check emptiness of just N or just A.

Subexponentials can be used to "locate" data and the promotion rule can be used to test selected locations for emptiness. These subexponentials provide linear logic specifications with enough checks on data to allow for a range of algorithms to be emulated *exactly* via (focused) proof search. We shall illustrate this claim by specifying a simple programming language, called BAG, containing loop instructions, conditionals and operations that insert into and delete from a multiset, which is powerful enough to specify complicated algorithms, such as *Dijkstra's algorithm* for finding the shortest distances in a positively weighted graph. We then show that for any BAG program there is a one-to-one correspondence between the set of its (partial) computations and the set of (open) focused proofs of its logic interpretation.

Since there is an exact correspondence between synthetic connectives in the logic and steps in the algorithmic language, we can vary the operational semantics of the algorithmic language by varying certain focusing-related features of the logic. In particular, by either inserting or removing *delay operators* into a logic specification, we can *package* more or fewer *operations* inside a synthetic connective. For example, reading two items from a multiset can be described as two algorithmic steps or as one algorithmic step.

In order to turn a logic specification into an algorithm, one must usually adopt an *interpreter* for the logic and then understand algorithmic nature of that interpreter. Top-down, depth-first interpreters are traditionally used to describe the algorithmic content of, for example, Horn clauses as Prolog programs. Other algorithmic rendering of logic clauses use bottom-up interpreters [Ganzinger 2001]. In this chapter, we shall not use any explicit interpreter for this role: instead, recent advances in proof theory will be used to organize proof search in algorithmically explicit but still fully declarative ways. In particular, we use the focused proof system for *SELL* for which possibly large sets of connectives can be treated as a single, monolithic, synthetic connective and where, in many situations, the inference rules related to such synthetic connectives can be applied in constant time. As a result, the particular nature of whichever interpreter one eventually uses for finding proofs can be largely eliminated when attempting to understand the algorithmic content of a logic specification.

The remainder of this chapter is structured as follows: in Section 7.2 we illustrate with a small example the increase of expressiveness obtained by using subexponentials. Section 7.3 contains some preliminary machinery we use in order to demonstrate how to specify algorithms in *SELLF*. We re-use the notion of *definitions* to specify arithmetic operations. We then explain how data is represented and stored by using subexponentials. Finally, we specialize the logic $SELL^{\text{m}}$ to allow for the creation of new locations. Section 7.4 introduces the syntax of the BAG programming language and shows how different intended semantics for it can be captured by using *SELLF* formulas. Sections 7.5 and 7.6 present some example algorithm specifications along with some comments about proof search complexity. Finally, in Section 7.7, we describe some related work, and in Section 7.8, we conclude by proposing some directions for future work.

7.2 Example: a minimal element of a multiset

Before we enter into the technical material, we illustrate with a small example the increase of expressiveness obtained by using subexponentials. Consider the nonempty multiset of natural numbers $\{m_1, \ldots, m_n\}$. Let $\langle\{\infty, l, k\}, \{k \preceq \infty, l \preceq \infty\}, \{\infty\}, \{\infty\}\rangle$ be a subexponential signature where l and k are not $\preceq$-comparable. Also, assume that all atoms are assigned negative polarity and let $\mathcal{K}$ be the indexed context where $\mathcal{K}[\infty]$ is the set

$$\{\ \exists x \exists y[l(x)^{\perp} \otimes l(y)^{\perp} \otimes (x \leq y) \otimes ?^{l} l(x)],\ \exists x[l(x)^{\perp} \otimes !^{k}\, min(x)]\ \},$$

$\mathcal{K}[k] = \emptyset$, and $\mathcal{K}[l] = \{l(m_1), \ldots, l(m_n)\}$. This context contains exactly two positive formulas and, hence, there are only two formulas on which to focus. We now derive in detail these two synthetic connectives.

Focusing on the first formula requires building the following derivation bottom-up:

$$\dfrac{\dfrac{\vdash \mathcal{K} : \cdot \Downarrow l(m_i)^\perp \otimes l(m_j)^\perp \otimes (m_i \leq m_j) \otimes ?^l l(m_i)}{\vdash \mathcal{K} : \cdot \Downarrow \exists x \exists y [l(x)^\perp \otimes l(y)^\perp \otimes (x \leq y) \otimes ?^l l(x)]}\,[2 \times \exists]}{\vdash \mathcal{K} : \cdot \Uparrow}\,[D_\infty]$$

Continuing this phase of the proof requires finding four indexed contexts such that $\mathcal{K}_1 \otimes \mathcal{K}_2 \otimes \mathcal{K}_3 \otimes \mathcal{K}_4[l] = \mathcal{K}[l]$ for all indexes l and such that the following four sequents $\vdash \mathcal{K}_1 : \cdot \Downarrow l(m_i)^\perp$, $\vdash \mathcal{K}_2 : \cdot \Downarrow l(m_j)^\perp$, $\vdash \mathcal{K}_3 : \cdot \Downarrow m_i \leq m_j$, and $\vdash \mathcal{K}_4 : \cdot \Downarrow ?^l l(m_i)$ are provable. The first two sequents are provable if and only if $\mathcal{K}_1[l] = \{l(m_i)\}$ and $\mathcal{K}_2[l] = \{l(m_j)\}$[1]. The third sequent is provable if m_i is less than or equal to m_j and $\mathcal{K}_3[l] = \{\}$. This means that $\mathcal{K}_4$ is the same as $\mathcal{K}$ except that $\mathcal{K}_4[l]$ is the multiset $\mathcal{K}[l]$ less the two distinct elements $l(m_i)$ and $l(m_j)$ (hence, $n > 1$). The remainder of this proof phase is necessarily of the form:

$$\dfrac{\dfrac{\dfrac{\vdash \mathcal{K}_4 +_l l(m_i) : \cdot \Uparrow \cdot}{\vdash \mathcal{K}_4 : \cdot \Uparrow ?^l l(m_i)}\,[?^l]}{\vdash \mathcal{K}_4 : \cdot \Downarrow ?^l l(m_i)}\,[R\Downarrow]$$

In other words, the synthetic connective arising from focusing on the first formula in the logic specification provides a proof of the sequent $\vdash \mathcal{K} : \cdot \Uparrow \cdot$ from the premise $\vdash \mathcal{K}' : \cdot \Uparrow \cdot$ exactly when $\mathcal{K}[l]$ contains at least 2 elements and $\mathcal{K}'$ is the same as $\mathcal{K}$ except that $\mathcal{K}'[l]$ results from $\mathcal{K}[l]$ by deleting one atom holding an integer greater than or equal to another integer in that multiset.

If we focus on the second formula, the resulting "macro" rule is built from the following "micro" rules.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \mathcal{K}_1 : \cdot \Downarrow l(m)^\perp}\,[I] \quad \dfrac{\dfrac{\vdash \mathcal{K}_2 : min(m) \Uparrow}{\vdash \mathcal{K}_2 : \cdot \Uparrow min(m)}\,[R\Uparrow]}{\vdash \mathcal{K}_2 : \cdot \Downarrow !^k min(m)}\,[!^k]}{\vdash \mathcal{K} : \cdot \Downarrow l(m)^\perp \otimes !^k min(m)}\,[\otimes]}{\vdash \mathcal{K} : \cdot \Downarrow \exists x [l(x)^\perp \otimes !^k min(x)]}\,[\exists]}{\vdash \mathcal{K} : \cdot \Uparrow \cdot}\,[D_\infty]$$

Here, $\mathcal{K}_1 \otimes \mathcal{K}_2 = \mathcal{K}$ and $\mathcal{K}_1[l] = \{l(m)\}$. Also, $\mathcal{K}_2[l]$ is empty, a fact guaranteed by the promotion rule and the fact that l and k are not $\preceq$-comparable. Thus, the corresponding synthetic connective provides a proof of the sequent $\vdash \mathcal{K} : \cdot \Uparrow$ from the premise $\vdash \mathcal{K}' : min(m) \Uparrow$ only when $\mathcal{K}[l]$ contains exactly one element (m) and $\mathcal{K}'$ is the result of setting the multiset $\mathcal{K}'[l]$ to the empty multiset.

The logic specification above clearly computes the minimal member of a multiset in a structured fashion: if the number of elements in the multiset (in location l) is one, then the minimum is found; and if the number of elements is more than one, then one element is discarded that does not affect the minimum. These two steps are described by focusing on different clauses. Notice that a proof using these clauses does not involve any backtracking from the point of synthetic connectives, while internal to the synthetic connective one might envision possible backtracking search (for example, to find m_i and m_j such that $m_i \leq m_j$).

[1] Remember that atoms, such as $l(m)$, are assigned negative polarity and, hence, $l(m)^\perp$ is assigned positive polarity. Moreover, only the initial rule can introduce a focused literal with positive polarity.

7.3 Preliminaries

In order to better illustrate some algorithms specifications in *SELLF*, we introduce the following machinery. First, we show how to specify arithmetic operations by using *definitions*. We then explain how we can represent *data structures* in *SELLF* by using subexponentials to *locate* such data structures. Finally, we propose a specialization of $SELL^{\mathbb{N}}$ that allows for the creation of new locations.

We also assume that the subexponential signature is $\Sigma = \langle I, \preceq, \mathcal{W}, \mathcal{C} \rangle$.

7.3.1 Including definitions and arithmetic operations

Several of the examples and algorithms that we consider in this chapter will need integers and some basic arithmetic operators on them. These all can be accommodated easily within *SELLF* in a purely "positive" setting. In particular, the arithmetic comparisons for integers, $\leq, <, =, \neq, >, \geq$, are available as binary predicates within *SELLF* by using definitions (see Section 6.2). For example the definition for $\leq$ is

$$x \leq y \quad \triangleq \quad [x = z] \oplus [\exists x' y' (x = s\, x') \otimes (y = s\, y') \otimes x' \leq y'].$$

The left disjunct specifies the base case when the first element is zero, denoted by the constant z, and the right disjunct specifies the recursion over the elements of the comparison. The other arithmetic comparisons are specified in a similar way.

If $\blacktriangledown$ denotes one of these relations, then the formula $m \blacktriangledown n$ is positive and provable instances of it are composed of exactly one positive phase and without the consumption of any formulas from the context. More formally, if $\mathcal{K}$ is an indexed context then $\vdash \mathcal{K} : \Gamma \Downarrow m \blacktriangledown n$ is provable if and only if m and n are integers that stand in the relation intended by $\blacktriangledown$ and $\Gamma \cup \mathcal{K}[I \setminus \mathcal{W}]$ is empty. We write $\widetilde{\blacktriangledown}$ to be the comparison that is the complement to the one denoted by $\blacktriangledown$: *e.g.*, $s \mathrel{\widetilde{\leq}} t$ is $s > t$.

We assume that basic integer addition and multiplication are also available as purely positive synthetic connectives. In particular, expressions such as $x \leq y + w$ are replaced by $\exists u.\mathit{plus}\ y\ w\ u \otimes x \leq u$, where $\mathit{plus}\ y\ w\ u$ denotes the relation between y and w and their sum u and is specified by the following definition:

$$\mathit{plus}\ y\ w\ u \quad \triangleq \quad [y = z \otimes w = u] \oplus [\exists y' u' (y = s\, y') \otimes (u = s\, u') \otimes \mathit{plus}\ y'\ w\ u'].$$

The left disjunct specifies the base case when the first element of the sum is zero, and the second disjunct specifies the recursion over the first element of the sum.

7.3.2 Representing Data Structures

As we described in Section 7.1, most of the dynamics of logic programming within classical and intuitionistic logic occurs within atomic formulas: thus, data structures are usually encoded as term structures so that they can appear within the scope of predicate constants. For example, a set of pairs $\{\langle x_1, y_1 \rangle, \ldots, \langle x_n, y_n \rangle\}$ can be encoded as the term $((x_1 :: y_1 :: \mathrm{nil}) :: \cdots :: (x_n :: y_n :: \mathrm{nil}) :: \mathrm{nil})$, where :: and nil are the non-empty and empty set constructors. In *SELLF*, it is possible to encode many data structures using multisets of formulas instead of terms. For example, the same set of pairs can be represented as

$$?^l \mathrm{rel}(x_1, y_1), \ldots, ?^l \mathrm{rel}(x_n, y_n)$$

in which the *subexponential* l provides a "location" for this data structure. Furthermore, the collection of formulas above encodes a *set* if $l \in \mathcal{C}$ or a *multiset* if $l \notin \mathcal{C}$.

In the rest of this chapter, we constrain indexed contexts as follows: for any subexponential $l \in I$, the multiset $\mathcal{K}[l]$ contains only atomic formulas and these are built with a predicate whose name is the same as l. Linking the predicate name of atomic formulas to their locations in this way is a convenience for the examples we shall consider. We also assume that all atoms used to encode data, *i.e.*, atoms in $\mathcal{K}[l]$ will be assigned negative polarity.

$$\frac{\mathcal{L} \cup \{loc\} \vdash \mathcal{K}, C}{\mathcal{L} \vdash \mathcal{K}, \Cap_l loc.C} \ [\Cap_l, \text{ provided } loc \text{ is a new location}]$$

$$\frac{\mathcal{L} \vdash \mathcal{K}, C[s/loc, \widehat{s}/\widehat{loc}]}{\mathcal{L} \vdash \mathcal{K}, \Cup_l loc.C} \ [\Cup_l, \text{ provided } s \in \mathcal{L}]$$

Figure 7.1: The introduction rules for $\Cap_l$ and $\Cup_l$. Here $\mathcal{L}$ is a set of locations.

7.3.3 Complements of locations

Since we will soon turn our attention to algorithm specifications in *SELLF*, we shall make two further restrictions in how we deploy *SELLF*.

First, we shall assume that all locations, l, except the special unbounded maximal location ∞, are bounded (that is $\mathcal{W} = \mathcal{C} = \{\infty\}$) and only related to ∞ (that is $l \preceq \infty$). Thus, data structures can be stored in different locations and no two locations will be considered sublocations.

Second, as the example above illustrated, testing that a given location l is empty required the promotion rule with a location k such that $k \not\preceq l$. To ensure that we have the ability to perform all such tests, we shall define the *complement* to the subexponential signature $\langle \{\infty\} \cup I, \preceq, \{\infty\}, \{\infty\} \rangle$ to be the signature $\langle \{\infty\} \cup I \cup \hat{I}, \hat{\preceq}, \{\infty\}, \{\infty\} \rangle$ where $\hat{I}$ is a copy of I containing elements of the form $\hat{l}$ whenever $l \in I$. The order relation $\preceq$ is extended with all pairs $\hat{l} \mathrel{\hat{\preceq}} k$ such that l and k are distinct members of I. Thus, in the complemented signature, $\hat{l}$ can be seen as a sublocation of all locations in I different from l. The promotion rule with the subexponential $!^{\hat{l}}$ succeeds only if the indexed context is empty at location l: all other locations need not be considered. We shall not "store" data in complemented locations: that is, $\mathcal{K}[\hat{l}]$ will always be empty.

7.3.4 Creation of new locations

Up to now, all locations are fixed throughout a proof. However, we have already proposed in Chapter 5 the proof system *SELL*$^{\Cap}$ that allows for the creation of new subexponential indexes. We propose here a specialization of such system, called *SELL*$^{\Cap_l}$, that, instead of considering general subexponential signatures, assumes only a set $\mathcal{L}$, containing all the bounded locations available to store data, and the existence of their complement locations, as discussed Subsection 7.3.3. It contains the connectives $\Cap_l$ and $\Cup_l$, whose introduction rules are given in Figure 7.1. The introduction rule for $\Cap_l$ creates a new location and its complement location, while the introduction rule for $\Cup_l$ instantiates all occurrences in C of loc by s and of $\widehat{loc}$ by $\widehat{s}$. As in the focused system for *SELL*$^{\Cap}$, we consider the connective $\Cap_l$ as asynchronous and $\Cup_l$ as synchronous.

7.4 Specifying Algorithms

There is a high-degree of "algorithmic context" in the description of synthetic connectives within *SELL*, especially once we made a few restrictions to that logic. In order to make the scope of such algorithmic specifications more evident, we present a small specification language that can be used to describe some *single-threaded* algorithms: while *multi-threaded* algorithmic specifications are possible in linear logic (see, for example, [Miller 1996]), we focus here on more traditional and determinant algorithmic specifications.

The following grammar introduces a high-level syntax for a small specification language we call Bag. We consider a fixed set of constants $\mathbb{C}$ that contains the natural numbers plus other tokens that we may need, such as *blue*, *red*, *etc.* We allow for two kinds of variables: members of $var \in \mathcal{V}$ denote variables over the first-order domain $\mathbb{C}$, while members of $K \in \mathcal{K}$ denote variables over programs

(continuations). To facilitate the construction of specifications in BAG, we introduce a new kind of variable $L \in \mathcal{L}$ for locations and introduce a set of constants $name \in \mathcal{N}$ for module names. The other syntactic classes can be defined as follow.

$$
\begin{array}{lll}
t & ::= c \in \mathbb{C} \mid var & tup ::= \langle t_1, \ldots, t_n \rangle \ (n \geq 0) \\
pat & ::= tup \mid \lambda var.pat & \\
cond_a & ::= t_1 \blacktriangledown t_2 & cond_l ::= \textbf{is\_empty}\ loc_b \\
cond & ::= cond_a \mid cond_l & \\
prog & ::= \textbf{load}\ tup\ loc\ prog \mid \textbf{unload}_i\ loc\ pat\ bprog & \\
 & \quad \mid \textbf{while}\ cond_a\ (\lambda K.prog)\ prog & \\
 & \quad \mid \textbf{loop}_i\ loc_b\ kprog\ prog \mid \textbf{new}\ loc\ \lambda L.prog & \\
 & \quad \mid \textbf{if}\ cond\ prog \mid prog \,[]\, prog \mid K \mid \textbf{end} & \\
bprog & ::= prog \mid \lambda var.bprog & \\
kprog & ::= \lambda K.prog \mid \lambda var.kprog & \\
lprog & ::= \lambda K.prog \mid \lambda L.lprog \mid \lambda var.lprog & \\
mod & ::= name \times lprog. &
\end{array}
$$

Conditions (tests) are of two kind: $cond_a$ are arithmetic tests (see Section 7.3.1) and $cond_l$ will be used to test if a given location is empty. The syntactic variable loc_b ranges over all bounded locations (here, all locations other than ∞ and complement locations). In the $\textbf{unload}_i$ (respectively, $\textbf{loop}_i$) instruction, we will also insist that *pat* and *bprog* (respectively, *kprog*) both have exactly i variables bindings. Moreover, when a module is used in a program, execution proceeds by computing the program resulting from performing the necessary beta reductions. Keeping with the single-threadedness of BAG, modules contain one and only one abstracted continuation variable.

The eight kinds of program types in BAG are described briefly as follow. (1) (**load** *tup loc prog*) inserts the tuple *tup* in the location *loc* and then continues with *prog*. (2) ($\textbf{unload}_i$ *loc pat bprog*) picks an element, $\langle t_1, \ldots, t_n \rangle$, from the location *loc* such that it matches with the term *pat* $t^1 \cdots t^i$ for some $t^j \in \{t_1, \ldots, t_n\}$ and then executes the program (*bprog* $t^1 \cdots t^i$). (3) (**while** $cond_a$ $(\lambda K.prog)$ *prog*) repeatedly applies $\lambda K.prog$ until the condition is not true; then *prog* is executed. (4) ($\textbf{loop}_i$ loc_b *pat bprog*) repeatedly executes ($\textbf{unload}_i$ loc_b $\lambda x_1 \ldots \lambda x_i.\langle x_1, \ldots, x_i \rangle$ *bprog*), where all $x_j \in \mathcal{V}$, until the location *loc* is empty. Intuitively, this loop is used to process all members of a location. (5) (**new** *loc* $\lambda L.prog$) creates a new location *loc* and then executes the program $(\lambda L.prog)loc$. (6) (**if** *cond prog*) executes *prog* if the condition *cond* holds. (7) ($prog_1$ [] $prog_2$) is an *alternative* instruction, where the computation proceeds to either $prog_1$ or $prog_2$. Lastly, (8) **end** ends the computation thread. Notice that this language is similar to Dijkstra's Guarded Command Language (GCL) [Dijkstra 1976]: in particular, the [] instruction is similar to GCL's *if* constructs, and the **while** and **loop** instructions are similar to GCL's *loop* constructs.

Our wish here is not to describe a new specification language but to highlight the algorithmic aspects already present within focused proof search in *SELL*. To this end, we show how the intended operational semantics of the BAG language can be specified by mapping it directly into *SELL* formulas. In particular, we will illustrate that the non-determinism that exists in an algorithmic description with, say, BAG, matches exactly the non-determinism in *SELLF*'s at the level of synthetic connectives. There is still more non-determinism one can imagine within proof search in *SELLF*, but those are contained *within* the construction macro-level inference rules. Later in Section 7.6, we shall discuss that in many cases, the construction of macro-level rules can, in fact, be done in constant time.

Being able to specify when a synthetic connective ends is critical for our claims about how focused proof search and algorithms in the BAG language relate. The two *delay operators* $\delta^-(\cdot)$ and $\delta^+(\cdot)$ can be used to replace a formula with a provably equivalent formula of a given polarity. In particular, $\delta^-(C)$ is negative no matter what polarity C is: it can be defined as $C \parr \bot$. Similarly, $\delta^+(C)$ is positive no matter what polarity C is: it can be defined as $C \otimes 1$.

$$
\begin{array}{lcl}
\textbf{load}\ \langle \bar{t} \rangle\ l\ prog & \triangleq & ?^{l} l(\bar{t}) \parr \delta^{+}(prog) \\
\textbf{unload}_i\ l\ pat\ bprog & \triangleq & l(pat\ v_1 \cdots v_i)^{\perp} \otimes [\delta^{-}(bprog\ v_1 \cdots v_i)] \\
\textbf{while}\ (t_1 \blacktriangledown t_2)\ (\lambda K.prog)\ prog & \triangleq & [(t_1 \blacktriangledown t_2) \otimes \delta^{-}((\lambda K.prog)\ (\textbf{while}\ (t_1 \blacktriangledown t_2)\ (\lambda K.prog)\ prog))] \\
 & & \quad \oplus [(t_1 \widetilde{\blacktriangledown} t_2) \otimes \delta^{-}(prog)] \\
\textbf{loop}_i\ l\ kprog\ prog & \triangleq & [l(v_1, \ldots, v_i)^{\perp} \otimes \delta^{-}((kprog\ v_1 \cdots v_i)\ (\textbf{loop}_i\ l\ kprog\ prog))] \\
 & & \quad \oplus !^{\hat{l}}(prog) \\
prog_1 \,[]\, prog_2 & \triangleq & prog_1 \oplus prog_2 \\
\textbf{if is\_empty}\ l\ prog & \triangleq & !^{\hat{l}}(prog) \\
\textbf{if}\ (t_1 \blacktriangledown t_2)\ prog & \triangleq & t_1 \blacktriangledown t_2 \otimes \delta^{-}(prog) \\
\textbf{new}\ loc\ \lambda L.prog & \triangleq & \Cap_l loc\ prog \\
\textbf{end} & \triangleq & \perp
\end{array}
$$

Figure 7.2: The definition clauses for specifying the execution of BAG programs.

The definition $\mathcal{D}$ in Figure 7.2 specifies a "proof theoretic" semantics of the BAG language. (For readability, we have suppressed writing the outermost universal quantifiers on these clauses.) The alternation of polarities, the use of the subexponential $!^{\hat{l}}$, and the placement of delays in this definition are particularly important to notice. For example, the meaning of the **load** command is given by using a negative formulas as its body: this command proceeds without needing any coordination with anything in the context, as illustrates the following derivation:

$$
\dfrac{\dfrac{\dfrac{\vdash \mathcal{K} +_l l(\bar{t}) : \cdot \Uparrow \delta^{+}(prog)}{\vdash \mathcal{K} : \cdot \Uparrow ?^{l} l(\bar{t}), \delta^{+}(prog)}\,[?^{l}]}{\vdash \mathcal{K} : \cdot \Uparrow ?^{l} l(\bar{t}) \parr \delta^{+}(prog)}\,[\parr]}{\vdash \mathcal{K} : \cdot \Uparrow \textbf{load}\ \langle \vec{t} \rangle\ prog}\,[def\ \Uparrow]
$$

Because of the positive delay $\delta^{+}(\cdot)$, it must be the case that the negative phase ends by performing $R\Uparrow$. Thus, this specification for **load** corresponds to the intended operation of loading *exactly one* tuple in a location.

All other instructions (except for **end** and **new**) are defined by positive formulas. In these cases, choices must be made and backtracking might be necessary inside a positive phase. For example, if one is focused on a **while** instruction then that focus continues on a formula of the form

$$\Downarrow [(t_1 \blacktriangledown t_2) \otimes \delta^{-}(C)] \oplus [(t_1 \widetilde{\blacktriangledown} t_2) \otimes \delta^{-}(D)]$$

At the "micro-rule level," proof search must pick between the two branches of the $\oplus$ and then determine which branch succeeds: at this level, some search may be required to compute the proper macro-step, but in the end, proof search will continue with either $\Uparrow C$ or with $\Uparrow D$ (the occurrences of $\delta^{-}(\cdot)$ forces the flip of $\Downarrow$ to $\Uparrow$): here, the choice is completely determined by the guards and this is reflected also with the "macro-level" inference rules.

Notice that there are no delays written into the definition of the [] operator since we wish that the choice provided by that operator is merged with choices in the instructions it accumulates. For example, the instructions

$$(\textbf{if}\ (x \leq y)\ prog_1) \,[]\, (\textbf{if}\ (\textbf{is\_empty}\ l)\ prog_2)$$

are equated, via the definition mechanism, to the formula

$$((x \leq y) \otimes \delta^{-}(prog_1)) \oplus \ !^{\hat{l}} prog_2.$$

This synthetic connective combines internally the test $x \leq y$ with the emptiness check of location l.

The correspondence between focused inference rules and algorithmic steps is precise: in particular, all partial proofs involving synthetic connectives match exactly the algorithmic steps that are possible. Thus, algorithmic steps that lead to failures are matched exactly with partial proofs that cannot be extended to complete proofs. As the behavior of an algorithm corresponds to the set of all its possible computation runs, this implies that the focused derivations obtained from Figure 7.2 capture exactly the behavior of BAG programs.

We now take the opportunity provided by this type of *full adequacy* to illustrate that by controlling the size of synthetic connectives, via the use of delays, we are able to capture different intended semantics for BAG and change the behavior of its programs. Consider the alternative operational semantics for alternation $[]$, where a step would correspond to only picking one of the non-deterministic choices and not executing the first command of the chosen program. We can capture such intended semantics by reducing the size of alternation's synthetic connective with the use of negative delays, as shown below:

$$prog_1 \ [] \ prog_2 \quad \triangleq \quad \delta^{-}(prog_1) \oplus \delta^{-}(prog_2).$$

Because of the extra negative delay operators, the positive phase must stop before applying the first instruction of the selected program. In this case, while the number of successful computation runs of a program does not change, the number of computation runs that fail might increase.

On the other hand, increasing the size of synthetic connectives, by removing delay operators, increases the amount of operations *packaged* in a synthetic connective, increasing, hence, the size of its corresponding transition step. These choices can also have deep consequences to the behavior of the system. Consider for example, a new definition for the **unload** instruction that does not contain a negative delay operator. In this case, one captures the intended semantics where all consecutive **unload** commands are performed in a single step. Since the non-determinism involved in picking the right tuples to unload is contained in the execution of a single transition step, the number of computation runs that succeed does not change, but the number of computation runs that fail might decrease.

Notice that since the **unload** and **load** operations are defined using dual connectives ($\otimes$ and $\parr$, respectively), they cannot be part of the same synthetic connective. Such a restriction on a synthetic connective (and on the associated algorithmic step) is sensible since the order in which one performs these operations can lead to different results.

7.5 Examples

The module *extractMin*, that extracts the minimum element from a multiset, is depicted in Figure 7.3. (For readability, the λ-abstractions associated with **unload** and **new** statements are elided, and we denote programs of the form $A\ (B\ C)$ as $(A;\ B\ C)$.) This module takes three locations, l_i, l_o, and *min*, and a continuation program *prog*. The module moves the minimum element of the multiset, located in l_i, to the location *min*, and moves its remaining elements to the location l_o.

The BAG program $\mathcal{P}^{\mathcal{G}}_{bp}$ in Figure 7.4 checks if a graph, $\mathcal{G}$, is bipartite. It takes as input three locations, for which all, except *ver*, are empty. Initially, all nodes are *gray* and later their color can change to *blue* or *red*. We use the location *ver* to store the nodes that are gray and the location *col* to store the nodes' color information. First, we create two auxiliary locations *pr* and *edges*. The first loop performs the initialization of the nodes' colors. Then, the second loop starts to traverse a new component of the graph, by picking any node from *ver*, assigning it the color *blue*, and inserting it

```
extractMin = λl_i λl_o λmin λprog.        //l_i - input location with a multiset of numbers;
                                          //l_o – output location with the remaining elements;
                                          //min – output location with the minimum element.
unload_2 l_i ⟨n, v⟩          //unload any element and set it as minimum
  load ⟨n, v⟩ min
  loop_2 l_i λn_1 λv_1 λlcont          //loop through the elements in l_i and update minimum element
    unload_2 min ⟨n_m, v_m⟩
      if (v_m ≤ v_1)
        load ⟨n_m, v_m⟩ min (load ⟨n_1, v_1⟩ l_o lcont)
      if (v_m > v_1)
        load ⟨n_1, v_1⟩ min (load ⟨n_m, v_m⟩ l_o lcont)
  prog   //no more elements in l_i; minimum element in min; and the remaining elements in l_o.
```

Figure 7.3: Extracting the minimum element

in the auxiliary location *pr*. The inner loop, that traverses through a component of the graph, starts by picking any node, *s*, in *pr*. It then invokes the module *getEdges* that loads the edges connected to *s* in the location *edges*. This module can be seen as a series of alternatives of **if** instructions, that checks the input node and loads accordingly the edges in a specified location. The third loop traverses through these edges. There are two alternatives, either *s* is *blue* or it is *red*. If it is blue, the program checks if all adjacent nodes, *adj*, are assigned the correct color (*red*), or assigns it the correct color and insert it in the location *pr*, or alternatively if *adj* is *blue* then the answer *no* is loaded in location *ans* and program finishes by proceeding to *prog*. A similar procedure is performed when *s* is *red*. If all nodes in *ver* are consumed then the graph is bipartite and the answer *yes* is loaded in the location *ans*.

The second example is the Dijkstra's algorithm that finds the shortest distance in a positively weighted graph, $\mathcal{G}$, which is specified by the program, $\mathcal{P}^{\mathcal{G}}_{dj}$, depicted in Figure 7.5. It contains two modules, the main module initializes the location *ver* by assigning the distance to all nodes to infinity, except the source node, *src*, whose distance is zero, and then calls the second module *dijkstra*. This module starts with two alternatives: if *ver* is empty, then the program ends with the shortest distances located in *dist*; or, it invokes the *extractMin* module, described before, to extract from *ver* the node, n_m, that has the minimum distance, which will be located in the auxiliary location *min*. The remaining nodes are transferred to the auxiliary location *ver*$'$. Then it adds n_m together with its distance in the location *dist*. Next, it invokes the module *getEdges* which loads in the auxiliary location *edges* all nodes adjacent to n_m with the associated cost of the edge. The program proceeds by looping among these edges and updating the distances of all nodes adjacent to n_m, in *ver*$'$, accordingly. Finally, the *dijkstra* module is called again but this giving as input the auxiliary location *ver*$'$, as the remaining nodes are now located there.

7.6 Complexity Analysis

The strong adequacy obtained for the encoding of Bag only ensures that any logic interpreter that searches for focused proofs by decomposing synthetic connectives will construct objects that correspond to computation runs of Bag programs. However, in order to analyze the complexity of algorithms, we must enter into implementation details. We now briefly propose an implementation that can, in many situations, compute in *constant time* if a synthetic connective can be used to help prove a given sequent. In particular, it is easy to show that it takes constant time to build a focusing

```
bipartite = λcolλverλansλprog.        //col - location with the colors of the nodes;
                                      //ver – location with the graph's unvisited vertices;
                                      //ans – output location with the answer yes or no.
new pr; new edges                     //create auxiliary locations.
loop1 ver λnλlcont                    //set node colors to gray.
  load ⟨n⟩ ver; load ⟨n, gray⟩ col lcont
loop1 ver λnλlcont1                   //pick a vertex, n, from a new component of the graph.
  unload0 col ⟨n, gray⟩               //n must be gray.
    load ⟨n, blue⟩ col; load ⟨n⟩ pr   //set n's color as blue, and store it in pr.
    loop1 pr λsλlcont2                //unload a vertex, s, that is in the same component.
      getEdges s edges                //loads the edges connected to s in the location edges.
      loop2 edges λsλadjλlcont3       //loop over the neighbors of s.
        unload0 col ⟨s, blue⟩; load ⟨s, blue⟩ col   //if the color of s is blue.
          unload0 col ⟨adj, red⟩      //and if the neighbor of s is red
            load ⟨adj, red⟩ col lcont3   //proceed.
          [] unload0 col ⟨adj, blue⟩  //if the neighbor of s is blue.
            load ⟨no⟩ ans prog        //graph not bipartite.
          [] unload0 col ⟨adj, gray⟩  //if the neighbor of s is gray,
              unload0 ver ⟨adj⟩⟩      //then it has not been yet visited, hence
                load ⟨adj, red⟩ col (load ⟨adj⟩ pr lcont3)   //assign it with the color red.
        [] unload0 col ⟨s, red⟩       //similar to the first alternative.
      lcont2
    lcont1
load ⟨yes⟩ ans prog                   //all nodes visited, hence the graph is bipartite.
```

Figure 7.4: Bipartite graph checking $\mathcal{P}^{\mathcal{G}}_{bp}$

phase with the body of the **load**, **while**, and **if** clauses, since arithmetic operations and comparisons can be assumed to be evaluated in constant time. Checking that the body of an alternative can be decomposed requires a search over all alternatives, which is bound by the size of the program, again a constant. The more interesting case involves determining if the body of an **unload** clause can be used since this clause involves pattern matching. In order to do pattern matching in constant time, we shall restrict tuples to be at most to arity 2. In that case, we represent the contents of such binary locations by using three linked hash-tables: one for when the pattern matching is on the first element; another hash-table when the pattern matching is on the second element; and finally the third hash-table is used when the pattern matching is on both elements. Hence, pattern matching is reduced to simple hash-table look-ups. Notice that one could do, in a similar fashion, constant time pattern matching even if tuples had arity greater than two: however that would come with a high cost in space.

Many algorithms, such as those described in Section 7.5, do not need to backtrack since all of their computation runs yield the same output. In the case of Dijkstra's algorithm, all of its computation runs end and have the same final output: namely the multiset containing the shortest distances. For these algorithms, we can use an interpreter that picks among several possible synthetic connectives and does not backtrack. Since decomposing a synthetic connective can take constant time, we can infer the complexity of an algorithm by counting the number of decide rules (the number of synthetic connectives) in a derivation that witnesses a complete computation run of an algorithm. For example, any derivation obtained from ($\mathcal{P}^{\mathcal{G}}_{bp}$ *col ver ans* **end**) where *ver* contains the nodes of the graph and all other locations are empty, contains $\mathcal{O}(|N| + |E|)$ decide rules, where $|N|$ and $|E|$ are the number

```
dijkstra = λverλdistλprog.
new ver′; new min; new edges                    //create auxiliary locations
if (is_empty ver) prog                          //finish if there are no more nodes to traverse
[] extractMin ver ver′ min                      //otherwise, call the extractMin module.
unload2 min ⟨n_m, c_m⟩; load ⟨n_m, c_m⟩ dist    //unload the minimum node, n_m.
  getEdges n_m edges                            //get the edges connected to n_m.
  loop2 edges λadjλdλlcont                      //update the distances of n_m's neighbors, adj.
    unload1 dist ⟨adj, c⟩                       //either, the shortest distance to adj is already computed,
      load ⟨adj, c⟩ dist lcont                  //then proceed;
    [] unload1 ver′ ⟨adj, c⟩                    //otherwise, check if there is a shorter path to adj.
      if (c ≤ d + c_m) (load ⟨adj, c⟩ ver′ lcont)
      [] if (c > d + c_m) (load ⟨adj, d + c_m⟩ ver′ lcont)
  dijkstra ver′ dist prog                       //call the dijkstra module.

main = λnodesλdistλsrcλprog.                    //nodes – location with the graph's nodes;
                                                //dist – location with the shortest distances.
                                                //src – name of the source node.
new ver                                         //create auxiliary location
loop1 nodes λnλlcont                            //set the distance of all nodes to ∞, except the source node.
  if (n ≠ src) (load ⟨n, ∞⟩ ver lcont)
  []
  if (n = src) (load ⟨s, 0⟩ ver lcont)
dijkstra ver dist prog                          //call the dijkstra module.
```

Figure 7.5: Dijkstra's algorithm $\mathcal{P}^{\mathcal{G}}_{dj}$.

of nodes and edges in a graph. Nodes are used at most three times and edges are used at most four times. Hence, the complexity of $\mathcal{P}^{\mathcal{G}}_{bp}$ is $\mathcal{O}(|N| + |E|)$.

7.7 Related Work

Various proposals for describing algorithms via rewriting multisets have been developed in the past. Probably one the earliest such proposals is the Gamma programming language [Banâtre 1996] although the even older specification language of Petri nets is also closely related to multiset rewriting. The Linda coordination model [Gelenter 1986] also makes use of primitive operations similar to those used in the manipulation of multisets. The close relationship between multiset-based computation and linear logic has been known and exploited for many years within early linear logic programming languages such as LinLog [Andreoli 1992], Lolli/Forum [Miller 1996], MSR [Cervesato 2001], and Lollimon [López 2005].

It is often difficult to directly relate the *search* for proofs (say, in a logic programming setting) to performing computations in a step-by-step, algorithmic sense. Probably the largest single problem in making this connection is the need to do backtracking during the search for proofs. Such backtracking might be acceptable if it can contained within "internal" and invisible processing steps, but it is unacceptable if such backtracking is done between "visible" steps, such as inputting and outputting. In this chapter, we tried to group possible backtracking points that are to be internal into single, macro-level inference steps: other non-deterministic choices are then left to the algorithm developer to organize appropriately.

Another approach for the treatment of backtracking is more global. One can describe computation

as a kind of forward chaining, generative model of computation. If one saturates a set of forward chaining rules with all possible consequences of a set of formulas, then failure to prove some atomic goal with respect to that saturation does not lead to backtracking. If some forward chaining is used but saturation is not done, then the failure to prove an atomic formula might be due to its not being provable or to not having accumulated this particular consequence yet: in the later case, one would need to backtrack and attempt to add more consequences. Saturation has been used in both the Gamma and the Lollimon setting as such a mechanism for dispelling backtracking. We have not pursued this approach here since we know of no proof theoretic treatment of saturation.

McAllester & Ganzinger [McAllester 2002, Ganzinger 2001, Ganzinger 2002] developed a style of algorithm specification, called "logical algorithms," that was inspired by bottom-up, logic programming specifications. In order to account for more algorithms, they moved beyond logic in order to incorporate the deletion of atomic formulas and the assignment of priorities to inference rule applications. Their framework was able to specify algorithms that efficiently solved problems from domains such as graph theory (*e.g.*, bipartite checking and the shortest distance problem), efficient data structures (*e.g.*, the Union/Find algorithm) and polymorphic type inference [McAllester 2003]. Simmons & Pfenning [Simmons 2008] revisited this style of logic specification and used linear logic inspired proof search to provide a sound foundation for the deletion of atomic facts.

Common to both the approaches by McAllester & Ganzinger and Simmons & Pfenning is the use of a bottom-up, generative interpreter that relies on saturation to control the scope of backtracking. By a careful and, at times, complex analysis of that particular interpreter, it is possible to guarantee efficient implementations for the specified logic programs.

There are two essential differences between our work and that on "logical algorithms." First, we have remained entirely within logic (in our case, linear logics with subexponentials) and have focused on not only soundness but also completeness. In fact, we have asked for more: we have insisted that the focused proofs that are built within that logic are in one-to-one correspondence with the steps of a simple algorithmic specification language. Second, we have not introduced the notion of an interpreter that directs search: in the "logical algorithm" papers, an algorithm's description is split between the logic specification *and* the interpreter. Here, there is no interpreter and the only structure given to proof search is that derived directly from focused proofs.

7.8 Conclusions and future works

In this chapter, we have investigated the increase of expressiveness resulted by adding subexponentials to linear logic. In particular, we have shown that a wide range of algorithms can be faithfully specified in *SELLF*. We used subexponentials to locate data structures, namely multiset of tuples, and proposed a simple programming language containing loop instructions and conditionals that manipulate these data structures. Then, we showed that several operational semantics for this programming language could be specified in *SELLF*, in such a way that there is a one-to-one correspondence between its (partial) computation runs and (open) derivations. We illustrated that more complicated algorithms, such as a bipartite checking algorithm and Dijsktra's shortest distance algorithm, can be expressed in such way.

We now point out some future research directions:

- *Implementation* - Still an implementation of a logic programming engine for *SELLF* is missing. One could imagine extending engines used for linear logic, *e.g.*, Lygon [Winikoff 1995], Lolli [Hodas 1994a], Lollimon [López 2005] and Forum [Miller 1996], to support subexponentials. It seems straightforward to extend the techniques proposed by Hodas *et al.* [Cervesato 1996, Hodas 1994b] to efficiently manage resources in *SELLF*: one must use, instead of one single bin with resources, several bins, each corresponding to a subexponential and associate to each bin the structural properties of the subexponential, weakening and contraction. One could use

some data structure to represent the preorder over the subexponentials, containing the number of elements in each bin, and when applying the non canonical bang rule, one would check if it is applicable by using such data structure.

- *Access Control* - For the security in computer systems, often, access control policies are established that determine if an agent's request should be processed. For example, an agent can only access a file in the system if it has authority over this file or if some authority is passed from another agent that has authority to do so. Recently, several specialized logics have been proposed as frameworks to specify and reason with such policies [Abadi 2003, Bowers 2007, Garg 2008]. It seems possible to use the preorder of the subexponentials in *SELLF* to specify, in a general framework, such control access policies: we use a global unbounded subexponential u for the general policy rules, and for each agent (or principal), A, we assign two subexponentials, ua and ba, such that ua is bounded and ba is affine and that $ba \preceq ua \preceq u$. We use the former subexponential, ua, to store agent A's persistent authority (or knowledge); for example, the authority of an agent to see the contents of its bank account (or the knowledge of its bank statement); and we use the latter subexponential ba to store agent A's consumable authority (or consumable resources) [Bowers 2007]; for example, an agent's consumable authority to spend money from its account. Notice that the subexponentials, ua and ba, of one agent are $\preceq$-unrelated to the subexponentials of another agent. Now, whenever we want to prove that an agent, A, knows or has the authority to perform some action, F, we try to prove the formula $!^{ba}F$. To prove this formula, we must first use the ba-bang rule, which forces that the formulas in the locations assigned to all other agents are weakened and only the formulas in the context ba, ua and u are used. On the other hand, if we want to add some knowledge or give some authority, F, to an agent, we can use the formula $?^{ua}F$ or $?^{ba}F$, depending if the knowledge or the authority is persistent or consumable. For example, if the agent B has authority over a file f, and it wants to give one-time file access permission to the agent A, we specify this operation by using the clause $aut(B, f, A)^{\perp} \otimes ?^{ba}per(f)$, where the predicate aut denotes that the agent B is giving access to the file f to the agent A. Since we add a consumable authority, specified by the ba-question mark, the agent A would only be authorized to access the file f once. Furthermore, it is easy to infer noninterference theorems: for example, if one adds new knowledge or authority to any other agent, the knowledge or authority of all other agents is unchanged.

  A direction for future work could be to check which properties or examples can be shown by using *SELLF* formulas to specify Access Control policies.

- *Concurrent Systems* - In this chapter, we investigated how to specify only sequential algorithms. Therefore, in all the clauses of the specifications, the macro-rule introducing a clause's synthetic connective has only one premise, and, therefore, there is only one continuation thread in Bag's instructions. One could imagine to specify algorithms that run in different computers by allowing clauses to have more than one premise, where each premise would represent a process in a different processor. This idea is similar to the *AND-concurrency* of Andreoli and Pareschi [Andreoli 1990b, Andreoli 1990a]. There is yet another direction one could pursue, namely by using, instead of single focused systems, such as *SELLF*, a *multifocused* system [Miller 2007b], where more than one formula can be focused on. In such system, one could allow more than one execution thread to be executed concurrently by focusing simultaneously on all defined formulas. Synthetic connectives in such system would correspond to transition steps where these threads are executed in parallel.

- *Playing with polarities* – Here, we have assumed that all atoms are given negative polarity. However, we know that more flexible polarity assignments could be given, without affecting provability, but affecting the shape of focused proofs. A direction for future work could be to investigate

the algorithmic content captured by such proofs where some atoms are assigned positive polarity. It seems possible, for example, to specify constraint systems, as in [Jagadeesan 2005], although without the one-to-one correspondence of partial computations and open derivations.

- *Chemical Abstract Programming* - Berry and Boudol [Berry 1990] proposed an alternative paradigm of programming, the chemical abstract programming. Consider initially that there is a multiset of elements and a sequence of sets of rewrite rules. The elements of the multiset *react* according to a set of rewrite rules, emulating a chemical process, and when no further reaction is possible, the next set of rules, in the sequence, is used to proceed with the computation. This approach was carried forward by Banâtre and Métayer in the GAMMA programming system [Banâtre 1996]. Later, Bruscoli and Guglielmi [Bruscoli 1996] used linear logic clauses in the *computation-as-proof-search* paradigm to give a purely logical account for GAMMA style programs, without the sequentiality operator. The problem to account for sequentiality was again the incapability of linear logic without subexponentials to specify when, only, certain linear resources are consumed. With the use of subexponentials, it seems now possible to specify the sequentiality operator in most of the examples shown by Banâtre and Métayer [Banâtre 1996], more specifically in those cases when the sequentiality is linked with the consumption of some specific resources. Another research project could be of investigating how far we can go with *SELLF*.

Chapter 8

Conclusions

We have exploited two non-canonical aspects of sequent calculus, namely the *polarity assignment in focused proofs* and the *linear logic exponentials*, by using the computation-as-proof-search paradigm in three different domains of computer science: *tabled deduction*, *logical frameworks* and *algorithmic specifications*.

At the end of each chapter, we have already discussed some pointers to future work directions. Now we just summarize the main contributions of this thesis.

- We provided a *proof theoretic explanation* for tabled deduction. More precisely, we have considered two cases: the first when tables contain only finite successes and the second when tables also contain finite failures. We then described, in Section 3.3, that we can incorporate a table into a proof by using multicut derivations and proposed, in Sections 3.6 and 3.9, focused proofs systems for the two cases considered. In some subsets of logic, namely hereditary Harrop formulas and the fragment of intuitionistic logic with fixed points used in [Tiu 2005], the only proofs and open derivations available in these systems are those that do not attempt to reprove tabled formulas. Later, we illustrated these results with several examples, including winning strategies, simulation, and a declarative specification for let-polymorphism type checking [Milner 1978].

- In Chapters 4 and 6, we demonstrated that linear logic can be used as a general framework for encoding proof systems for minimal, intuitionistic and classical logics. The proposed theories are such that the focused derivations obtained from them and the derivations of the encoded proof systems are in one-to-one correspondence. We also showed, in Chapter 4, that one can directly infer relative completeness results between different encoded proof systems; for example, that sequent calculus and natural deduction systems for intuitionistic logic prove the same theorems.

- We investigated, in Chapter 5, the proof theory for *subexponentials* and proposed different focused systems containing these non-canonical exponentials. First in Subsection 5.2.1, we considered the case when relevant formulas are not allowed and then in Subsection 5.2.2, we proposed two focused proof systems for logics containing relevant formulas. Later in Section 5.3, we proposed an extension to linear logic that allows for the creation and modification of subexponential indexes.

- Our last contribution consisted in illustrating what kinds of algorithms can be faithfully specified by using subexponentials. We use subexponentials to locate multiset of tuples and show, in Chapter 7, that the operational semantics of a simple programming language, containing loop instructions, conditionals and operations that include to and delete from a multiset, can be captured by using linear logic with subexponentials. Finally, we illustrate this result with several graph algorithms, such as *Dijkstra's algorithm* for finding the shortest distances in a positively weighted graph, and an algorithm for checking if a graph is bipartite.

Appendix A

Appendix

A.1 Natural deduction rules and their linear logic encodings

We list below several examples of how natural deduction rules are accounted for by focused deduction in linear logic. The following correspondences can be used to prove Proposition 4.7. In the derivations below, $\mathcal{K} = \mathcal{L} \cup \{Str_L, Id_1, Id_2\} \cup \lfloor \Gamma \rfloor$ and all $\lceil \cdot \rceil$ given negative polarity and all $\lfloor \cdot \rfloor$ are given positive polarity.

$$\frac{}{\Gamma, C \vdash C \downarrow}\,[\mathrm{I}] \quad \rightsquigarrow \quad \frac{\dfrac{}{\vdash \mathcal{K}, \lfloor C \rfloor : \lfloor C \rfloor^{\perp} \Downarrow \lfloor C \rfloor}\,[I_1]}{\vdash \mathcal{K}, \lfloor C \rfloor : \lfloor C \rfloor^{\perp} \Uparrow}\,[D_2]$$

$$\frac{\Gamma \vdash A \downarrow}{\Gamma \vdash A \uparrow}\,[\mathrm{M}] \quad \rightsquigarrow \quad \frac{\dfrac{\dfrac{\vdash \mathcal{K} : \lfloor C \rfloor^{\perp} \Uparrow}{\vdash \mathcal{K} :\Downarrow \lfloor C \rfloor^{\perp}}\,[R\Downarrow, R\Uparrow] \quad \dfrac{}{\vdash \mathcal{K} : \lceil C \rceil \Downarrow \lceil C \rceil^{\perp}}\,[I_1]}{\vdash \mathcal{K} : \lceil C \rceil \Downarrow \lfloor C \rfloor^{\perp} \otimes \lceil C \rceil^{\perp}}\,[\otimes]}{\vdash \mathcal{K} : \lceil C \rceil \Uparrow}\,[D_2, \exists]$$

$$\frac{\Gamma \vdash A \uparrow}{\Gamma \vdash A \downarrow}\,[\mathrm{S}] \quad \rightsquigarrow \quad \frac{\dfrac{\dfrac{}{\vdash \mathcal{K} : \lfloor C \rfloor^{\perp} \Downarrow \lfloor C \rfloor}\,[I_1] \quad \dfrac{\vdash \mathcal{K} : \lceil C \rceil \Uparrow}{\vdash \mathcal{K} :\Downarrow\, !\lceil C \rceil}\,[!, R\Uparrow]}{\vdash \mathcal{K} : \lfloor C \rfloor^{\perp} \Downarrow \lfloor C \rfloor \otimes\, !\lceil C \rceil}\,[\otimes]}{\vdash \mathcal{K} : \lfloor C \rfloor^{\perp} \Uparrow}\,[D_2, \exists]$$

$$\frac{\Gamma \vdash F \uparrow \quad \Gamma \vdash G \uparrow}{\Gamma \vdash F \wedge G \uparrow}\,[\wedge I] \quad \rightsquigarrow$$

$$\frac{\dfrac{\dfrac{}{\vdash \mathcal{K} : \lceil F \wedge G \rceil \Downarrow \lceil F \wedge G \rceil^{\perp}}\,[I_1] \quad \dfrac{\vdash \mathcal{K} : \lceil F \rceil \Uparrow}{\vdash \mathcal{K} :\Downarrow \lceil F \rceil}\,[R\Downarrow, R\Uparrow] \quad \dfrac{\vdash \mathcal{K} : \lceil G \rceil \Uparrow}{\vdash \mathcal{K} :\Downarrow \lceil G \rceil}\,[R\Downarrow, R\Uparrow]}{\vdash \mathcal{K} : \lceil F \wedge G \rceil \Downarrow \lceil F \wedge G \rceil^{\perp} \otimes \lceil F \rceil \otimes \lceil G \rceil}\,[2 \times \otimes]}{\vdash \mathcal{K} : \lceil F \wedge G \rceil \Uparrow}\,[D_2, 2 \times \exists]$$

$$\frac{\Gamma \vdash F \wedge G \downarrow}{\Gamma \vdash F \downarrow}\,[\wedge E] \quad \rightsquigarrow \quad \frac{\dfrac{\dfrac{\vdash \mathcal{K} : \lfloor F \wedge G \rfloor^{\perp} \Uparrow}{\vdash \mathcal{K} :\Downarrow \lfloor F \wedge G \rfloor^{\perp}}\,[R\Downarrow, R\Uparrow] \quad \dfrac{}{\vdash \mathcal{K} : \lfloor F \rfloor^{\perp} \Downarrow \lfloor F \rfloor}\,[I_1]}{\vdash \mathcal{K} : \lceil F \wedge G \rceil \Downarrow \lfloor F \wedge G \rfloor^{\perp} \otimes (\lfloor F \rfloor \oplus \lfloor G \rfloor)}\,[\otimes, \oplus_1]}{\vdash \mathcal{K} : \lfloor F \rfloor^{\perp} \Uparrow}\,[D_2, 2 \times \exists]$$

$$\frac{\Gamma \vdash A_i \uparrow}{\Gamma \vdash A_1 \vee A_2 \uparrow}\,[\vee I, i \in \{1, 2\}] \quad \rightsquigarrow$$

$$\frac{\dfrac{\dfrac{}{\vdash \mathcal{K} : \lceil A_1 \vee A_2 \rceil \Downarrow \lceil A_1 \vee A_2 \rceil^{\perp}}\,[I_1] \quad \dfrac{\vdash \mathcal{K} : \lceil A_i \rceil \Uparrow}{\vdash \mathcal{K} :\Downarrow \lfloor A_1 \rfloor \oplus \lceil A_2 \rceil}\,[\oplus_i, R\Downarrow, R\Uparrow]}{\vdash \mathcal{K} : \lceil A_1 \vee A_2 \rceil \Downarrow \lceil A_1 \vee A_2 \rceil^{\perp} \otimes (\lfloor A_1 \rfloor \oplus \lceil A_2 \rceil)}\,[2 \times \otimes]}{\vdash \mathcal{K} : \lceil A_1 \vee A_2 \rceil \Uparrow}\,[D_2, 2 \times \exists]$$

$$\dfrac{\Gamma \vdash A \vee B \downarrow \quad \Gamma, A \vdash C \uparrow(\downarrow) \quad \Gamma, A \vdash C \uparrow(\downarrow)}{\Gamma \vdash C \uparrow(\downarrow)}\,[\vee E] \quad \leftrightsquigarrow$$

$$\dfrac{\dfrac{\dfrac{\vdash \mathcal{K} : \lfloor A \vee B \rfloor^{\perp} \Uparrow}{\vdash \mathcal{K} :\Downarrow\, !\lfloor A \vee B \rfloor^{\perp}}\,[!, R\Uparrow] \quad \dfrac{\dfrac{\vdash \mathcal{K}, \lfloor A \rfloor : \lceil C \rceil\,(\lfloor C \rfloor^{\perp}) \Uparrow}{\vdash \mathcal{K} : \lceil C \rceil\,(\lfloor C \rfloor^{\perp}) \Uparrow ?\lfloor A \rfloor}\,[?] \quad \dfrac{\vdash \mathcal{K}, \lfloor B \rfloor : \lceil C \rceil\,(\lfloor C \rfloor^{\perp}) \Uparrow}{\vdash \mathcal{K} : \lceil C \rceil\,(\lfloor C \rfloor^{\perp}) \Uparrow ?\lfloor B \rfloor}\,[?]}{\vdash \mathcal{K} : \lceil C \rceil\,(\lfloor C \rfloor^{\perp}) \Downarrow ?\lfloor A \rfloor \,\&\, ?\lfloor B \rfloor}\,[R\Downarrow, \&]}{\vdash \mathcal{K} : \lceil C \rceil\,(\lfloor C \rfloor^{\perp}) \Downarrow\, !\lfloor A \vee B \rfloor^{\perp} \otimes (?\lfloor A \rfloor \,\&\, ?\lfloor B \rfloor)}\,[\otimes]}{\vdash \mathcal{K} : \lceil C \rceil\,(\lfloor C \rfloor^{\perp}) \Uparrow}\,[D_2, 2 \times \exists]$$

$$\dfrac{\Gamma, A \vdash B \uparrow}{\Gamma \vdash A \supset B \uparrow}\,[\supset I] \quad \leftrightsquigarrow$$

$$\dfrac{\dfrac{\dfrac{}{\vdash \mathcal{K} : \lceil A \supset B \rceil \Downarrow \lceil A \supset B \rceil^{\perp}}\,[I_1] \quad \dfrac{\dfrac{\vdash \mathcal{K}, \lfloor A \rfloor : \lceil B \rceil \Uparrow}{\vdash \mathcal{K} :\Uparrow ?\lfloor A \rfloor, \lceil B \rceil}\,[?, R\Uparrow]}{\vdash \mathcal{K} :\Downarrow ?\lfloor A \rfloor \parr \lceil B \rceil}\,[R\Downarrow, \parr]}{\vdash \mathcal{K} : \lceil A \supset B \rceil \Downarrow \lceil A \supset B \rceil^{\perp} \otimes (?\lfloor A \rfloor \parr \lceil B \rceil)}\,[\otimes]}{\vdash \mathcal{K} : \lceil A \supset B \rceil \Uparrow}\,[D_2, 2 \times \exists]$$

$$\dfrac{\Gamma \vdash A \supset B \downarrow \quad \Gamma \vdash A \uparrow}{\Gamma \vdash B \downarrow}\,[\supset E] \quad \leftrightsquigarrow$$

$$\dfrac{\dfrac{\dfrac{\vdash \mathcal{K} : \lfloor A \supset B \rfloor^{\perp} \Uparrow}{\vdash \mathcal{K} :\Downarrow \lfloor A \supset B \rfloor^{\perp}}\,[R\Downarrow, R\Uparrow] \quad \dfrac{\vdash \mathcal{K} : \lceil A \rceil \Uparrow}{\vdash \mathcal{K} :\Downarrow\, !\lceil A \rceil}\,[!, R\Uparrow] \quad \dfrac{}{\vdash \mathcal{K} : \lfloor B \rfloor^{\perp} \Downarrow \lfloor B \rfloor}\,[I_1]}{\vdash \mathcal{K} : \lfloor B \rfloor^{\perp} \Downarrow \lfloor A \supset B \rfloor^{\perp} \otimes (!\lceil A \rceil \otimes \lfloor B \rfloor)}\,[2 \times \otimes]}{\vdash \mathcal{K} : \lfloor B \rfloor^{\perp} \Uparrow}\,[D_2, 2 \times \exists]$$

$$\dfrac{}{\Gamma \vdash t \uparrow}\,[tI] \quad \leftrightsquigarrow \quad \dfrac{\dfrac{\dfrac{}{\vdash \mathcal{K} : \lceil t \rceil \Downarrow \lceil t \rceil^{\perp}}\,[I_1] \quad \dfrac{}{\vdash \mathcal{K} :\Downarrow \top}\,[R\Downarrow, \top]}{\vdash \mathcal{K} : \lceil t \rceil \Downarrow \lceil t \rceil^{\perp} \otimes \top}\,[\otimes]}{\vdash \mathcal{K} : \lceil t \rceil \Uparrow}\,[D_2]$$

$$\dfrac{\Gamma \vdash \perp \downarrow}{\Gamma \vdash C \uparrow}\,[\perp E] \quad \leftrightsquigarrow \quad \dfrac{\dfrac{\dfrac{}{\vdash \mathcal{K}, \lfloor \Gamma \rfloor : \lceil C \rceil \Downarrow \lceil C \rceil^{\perp}}\,[I_1] \quad \dfrac{\vdash \mathcal{K}, \lfloor \Gamma \rfloor : \cdot \Uparrow}{\vdash \mathcal{K}, \lfloor \Gamma \rfloor :\Downarrow \perp}\,[R\Downarrow, \perp]}{\vdash \mathcal{K}, \lfloor \Gamma \rfloor : \lceil C \rceil \Downarrow \lceil C \rceil^{\perp} \otimes \perp}\,[\otimes]}{\vdash \mathcal{K}, \lfloor \Gamma \rfloor : \lceil C \rceil \Uparrow}\,[D_2, \exists]$$

$$\dfrac{\Gamma \vdash A\{c/x\} \uparrow}{\Gamma \vdash \forall x\, A \uparrow}\,[\forall I] \quad \leftrightsquigarrow \quad \dfrac{\dfrac{\dfrac{}{\vdash \mathcal{K} : \lceil \forall x\, A \rceil \Downarrow \lceil \forall x\, A \rceil^{\perp}}\,[I_1] \quad \dfrac{\vdash \mathcal{K} : \lceil A\{c/x\} \rceil \Uparrow}{\vdash \mathcal{K} :\Downarrow \forall x \lceil A \rceil}\,[R\Downarrow, \forall, R\Uparrow]}{\vdash \mathcal{K} : \lceil \forall x\, A \rceil \Downarrow \lceil \forall x\, A \rceil^{\perp} \otimes \forall x \lceil A \rceil}\,[\otimes]}{\vdash \mathcal{K} : \lceil \forall x\, A \rceil \Uparrow}\,[D_2, \exists]$$

$$\dfrac{\Gamma \vdash \forall x\, A \downarrow}{\Gamma \vdash A\{t/x\} \downarrow}\,[\forall E] \quad \leftrightsquigarrow$$

$$\dfrac{\dfrac{\dfrac{\vdash \mathcal{K} : \lfloor \forall x\, A \rfloor^{\perp} \Uparrow}{\vdash \mathcal{K} :\Downarrow \lfloor \forall x\, A \rfloor^{\perp}}\,[R\Downarrow, \forall, R\Uparrow] \quad \dfrac{}{\vdash \mathcal{K} : \lfloor A\{t/x\} \rfloor^{\perp} \Downarrow \lfloor A\{t/x\} \rfloor}\,[I_1]}{\vdash \mathcal{K} : \lfloor A\{t/x\} \rfloor^{\perp} \Downarrow \lfloor \forall x\, A \rfloor^{\perp} \otimes \lfloor A\{t/x\} \rfloor}\,[\otimes]}{\vdash \mathcal{K} : \lfloor A\{t/x\} \rfloor^{\perp} \Uparrow}\,[D_2, \exists]$$

The pairing for the $\exists I$ and $\exists E$ rules are similar.

Bibliography

[Abadi 2003] Martín Abadi. *Logic in Access Control.* In LICS, pages 228–. IEEE Computer Society, 2003. 139

[Anderson 1975] Alan R. Anderson and Nuel D. Belnap. Entailment: The logic of relevance and necessity. Princeton University Press, Princeton, NJ, 1975. 23

[Andreoli 1990a] J.-M. Andreoli and R. Pareschi. *Linear Objects: Logical Processes with Built-In Inheritance.* In Proceeding of the Seventh International Conference on Logic Programming, Jerusalem, May 1990. 139

[Andreoli 1990b] Jean-Marc Andreoli and Remo Pareschi. *LO and Behold! Concurrent Structured Processes.* In OOPSLA/ECOOP, pages 44–56, 1990. 139

[Andreoli 1992] Jean-Marc Andreoli. *Logic Programming with Focusing Proofs in Linear Logic.* J. of Logic and Computation, vol. 2, no. 3, pages 297–347, 1992. 1, 13, 15, 16, 88, 137

[Baelde 2007a] David Baelde, Andrew Gacek, Dale Miller, Gopalan Nadathur and Alwen Tiu. *The Bedwyr system for model checking over syntactic expressions.* In Frank Pfenning, editeur, 21th Conference on Automated Deduction (CADE), numéro 4603 de LNAI, pages 391–397. Springer, 2007. 42, 43, 57

[Baelde 2007b] David Baelde and Dale Miller. *Least and greatest fixed points in linear logic.* In N. Dershowitz and A. Voronkov, editeurs, International Conference on Logic for Programming and Automated Reasoning (LPAR), volume 4790 of *LNCS*, pages 92–106, 2007. 41, 42

[Baelde 2008] David Baelde. *A linear approach to the proof-theory of least and greatest fixed points.* PhD thesis, Ecole Polytechnique, December 2008. 41, 42

[Banâtre 1996] Jean-Pierre Banâtre and Daniel Le Métayer. *Gamma and the chemical reaction model: ten years after.* In Coordination programming: mechanisms, models and semantics, pages 3–41. World Scientific Publishing, IC Press, 1996. 137, 140

[Berry 1990] G. Berry and G. Boudol. *The Chemical Abstract Machine.* In 17th ACM Symp. on Principles of Programming Languages, pages 81–94, 1990. 140

[Bowers 2007] Kevin D. Bowers, Lujo Bauer, Deepak Garg, Frank Pfenning and Michael K. Reiter. *Consumable credentials in logic-based access-control systems.* In NDSS'07, 2007. 139

[Braüner 1996] Torben Braüner and Valeria de Paiva. *Cut-Elimination for Full Intuitionistic Linear Logic.* Technical Report BRICS-RS-96-10, University of Aarhus, 1996. 88, 99

[Bruscoli 1996] Paola Bruscoli and Alessio Guglielmi. *A Linear Logic View of Gamma Style Computations as Proof Searches.* In Jean-Marc Andreoli, Chris Hankin and Daniel Le Métayer, editeurs, Coordination Programming: Mechanisms, Models and Semantics. Imperial College Press, 1996. 140

[Cervesato 1996] Iliano Cervesato, Joshua Hodas and Frank Pfenning. *Efficient Resource Management for Linear Logic Proof Search.* In Roy Dyckhoff, Heinrich Herre and Peter Schroeder-Heister, editeurs, 7th Workshop on Extensions to Logic Programming, LNAI, pages 28–30, Leipzig, Germany, March 1996. Springer-Verlag. 138

[Cervesato 2001] Iliano Cervesato. *Typed MSR: Syntax and Examples.* In MMMACNS: International Workshop on Methods, Models and Architectures for Network Security, volume 2052 of *LNCS*, pages 159–177. Springer, 2001. 137

[Chaudhuri 2006] Kaustuv Chaudhuri. *The Focused Inverse Method for Linear Logic.* PhD thesis, Carnegie Mellon University, December 2006. Technical report CMU-CS-06-162. 102

[Chaudhuri 2008a] Kaustuv Chaudhuri, Dale Miller and Alexis Saurin. *Canonical Sequent Proofs via Multi-Focusing.* In Juhani Karhumäki and Luke Ong, editeurs, IFIP International Conference on Theoretical Computer Science, September 2008. To appear. 82

[Chaudhuri 2008b] Kaustuv Chaudhuri, Frank Pfenning and Greg Price. *A Logical Characterization of Forward and Backward Chaining in the Inverse Method.* J. of Automated Reasoning, vol. 40, no. 2-3, pages 133–177, March 2008. 19

[Chaudhuri 2009] Kaustuv Chaudhuri. On the expressivity of two refinements of multiplicative exponential linear logic. Submitted, 2009. 103

[Church 1940] Alonzo Church. *A Formulation of the Simple Theory of Types.* J. of Symbolic Logic, vol. 5, pages 56–68, 1940. 5

[Ciabattoni 2008] Agata Ciabattoni, Nikolaos Galatos and Kazushige Terui. *From axioms to analytic rules in nonclassical logics.* In 23th Symp. on Logic in Computer Science, pages 229–240. IEEE Computer Society Press, 2008. 81

[D'Agostino 1994] Marcello D'Agostino and Marco Mondadori. *The Taming of the Cut. Classical Refutations with Analytic Cut.* J. Log. Comput., vol. 4, no. 3, pages 285–319, 1994. 3, 4, 60, 77

[Danos 1993] Vincent Danos, Jean-Baptiste Joinet and Harold Schellinx. *The Structure of Exponentials: Uncovering the Dynamics of Linear Logic Proofs.* In Georg Gottlob, Alexander Leitsch and Daniele Mundici, editeurs, Kurt Gödel Colloquium, volume 713 of *LNCS*, pages 159–171. Springer, 1993. 2, 4, 13, 85, 86, 103

[Danos 1995] V. Danos, J.-B. Joinet and H. Schellinx. *LKT and LKQ: sequent calculi for second order logic based upon dual linear decompositions of classical implication.* In J.-Y. Girard, Y. Lafont and L. Regnier, editeurs, Advances in Linear Logic, numéro 222 de London Mathematical Society Lecture Note Series, pages 211–224. Cambridge University Press, 1995. 19

[Dijkstra 1976] Edsger W. Dijkstra. A discipline of programming. Prentice-Hall, 1976. 132

[Dyckhoff 2006] R. Dyckhoff and S. Lengrand. *LJQ: a strongly focused calculus for intuitionistic logic.* In A. Beckmann et al, editeur, Computability in Europe 2006, volume 3988 of *LNCS*, pages 173–185. Springer, 2006. 3, 4, 106, 113, 124

[Dyckhoff 2007] Roy Dyckhoff and Stephane Lengrand. *Call-by-Value λ-calculus and LJQ.* J. of Logic and Computation, vol. 17, no. 6, pages 1109–1134, 2007. 19

[Felty 1988] Amy Felty and Dale Miller. *Specifying theorem provers in a higher-order logic programming language.* In Ninth International Conference on Automated Deduction, pages 61–80, Argonne, IL, May 1988. Springer-Verlag. 2, 59, 81

[Gacek 2008] Andrew Gacek. *The Abella Interactive Theorem Prover (System Description).* In Fourth International Joint Conference on Automated Reasoning, 2008. Available from http://arxiv.org/abs/0803.2305. To appear in IJCAR. 42

[Ganzinger 2001] Harald Ganzinger and David A. McAllester. *A New Meta-complexity Theorem for Bottom-Up Logic Programs.* In Rajeev Goré, Alexander Leitsch and Tobias Nipkow, editeurs, Automated Reasoning, First International Joint Conference (IJCAR), volume 2083 of *Lecture Notes in Computer Science*, pages 514–528. Springer, 2001. 128, 138

[Ganzinger 2002] H. Ganzinger and D. McAllester. *Logical Algorithms.* In Proc. ICLP 2002, volume 2401 of *LNCS*, pages 209–223. Springer-Verlag, 2002. 138

[Garg 2008] Deepak Garg and Martín Abadi. *A Modal Deconstruction of Access Control Logics.* In Roberto M. Amadio, editeur, FoSSaCS, volume 4962 of *Lecture Notes in Computer Science*, pages 216–230. Springer, 2008. 139

[Gelenter 1986] David Gelenter. *Generative communication in Linda.* ACM Transactions on Programming Languages and Systems, vol. 7, no. 1, pages 80–112, 1986. 137

[Gentzen 1969] Gerhard Gentzen. *Investigations into Logical Deductions.* In M. E. Szabo, editeur, The Collected Papers of Gerhard Gentzen, pages 68–131. North-Holland, Amsterdam, 1969. 1, 3, 4, 5, 6, 24, 60, 67, 105

[Girard 1987] Jean-Yves Girard. *Linear Logic.* Theoretical Computer Science, vol. 50, pages 1–102, 1987. 2, 10, 13

[Girard 1989] Jean-Yves Girard, Paul Taylor and Yves Lafont. Proofs and types. Cambridge University Press, 1989. 5

[Girard 1991] Jean-Yves Girard. *On the unity of logic.* Rapport technique 26, Université Paris VII, June 1991. 103, 118

[Girard 1992] Jean-Yves Girard. A fixpoint theorem in linear logic. An email posting to the mailing list linear@cs.stanford.edu, February 1992. 41

[Girard 2006] Jean-Yves Girard. Le point aveugle: Cours de logique: Tome 1, vers la perfection. Hermann, 2006. 62

[Gödel 1958] Kurt Gödel. *Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes.* Dialectica 12, pages 280–287, 1958. 102

[Harper 1993] Robert Harper, Furio Honsell and Gordon Plotkin. *A Framework for Defining Logics.* Journal of the ACM, vol. 40, no. 1, pages 143–184, 1993. 2, 59, 81

[Herbelin 1995] Hugo Herbelin. *Séquents qu'on calcule: de l'interprétation du calcul des séquents comme calcul de lambda-termes et comme calcul de stratégies gagnantes.* PhD thesis, Université Paris 7, 1995. 3, 4, 106, 113, 124

[Hernest 2008] Mircea-Dan Hernest and Paulo Oliva. *Hybrid Functional Interpretations.* In Arnold Beckmann, Costas Dimitracopoulos and Benedikt Löwe, editeurs, CiE, volume 5028 of *Lecture Notes in Computer Science*, pages 251–260. Springer, 2008. 102

[Hodas 1994a] Joshua Hodas and Dale Miller. *Logic Programming in a Fragment of Intuitionistic Linear Logic.* Information and Computation, vol. 110, no. 2, pages 327–365, 1994. 81, 138

[Hodas 1994b] Joshua S. Hodas. *Logic Programming in Intuitionistic Linear Logic: Theory, Design, and Implementation.* PhD thesis, University of Pennsylvania, Department of Computer and Information Science, May 1994. 102, 138

[Howard 1980] William A. Howard. *The formulae-as-type notion of construction, 1969.* In J. P. Seldin and R. Hindley, editeurs, To H. B. Curry: Essays in Combinatory Logic, Lambda Calculus, and Formalism, pages 479–490. Academic Press, New York, 1980. 1

[Jagadeesan 2005] Radha Jagadeesan, Gopalan Nadathur and Vijay Saraswat. *Testing concurrent systems: An interpretation of intuitionistic logic.* In FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science, volume 3821 of *LNCS*, Hyderabad, India, 2005. Springer. 2, 32, 124, 140

[Kleene 1968] S. C. Kleene. Mathematical logic. Wiley and Sons, 1968. 5

[Kreisel 1959] Georg Kreisel. *Interpretation of Analysis by Means of Constructive Functionals of Finite Types.* In Constructivity in Mathematics, pages 101–128, 1959. 102

[Liang 2007] Chuck Liang and Dale Miller. *Focusing and Polarization in Intuitionistic Logic.* In J. Duparc and T. A. Henzinger, editeurs, CSL 2007: Computer Science Logic, volume 4646 of *LNCS*, pages 451–465. Springer, 2007. 2, 3, 19, 25, 32, 106, 116, 124

[Liang 2008] Chuck Liang and Dale Miller. Focusing and polarization in linear, intuitionistic, and classical logics. Accepted to *Theoretical Computer Science*, 2008. 2, 3, 4, 20, 25, 27, 57, 106, 116, 120, 124

[Liang 2009] Chuck Liang and Dale Miller. A unified sequent calculus for focused proofs. Accepted to LICS, 2009. 103

[López 2005] Pablo López, Frank Pfenning, Jeff Polakow and Kevin Watkins. *Monadic concurrent linear logic programming.* In Pedro Barahona and Amy P. Felty, editeurs, Proceedings of the 7th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP), pages 35–46. ACM, 2005. 137, 138

[Maehara 1954] S. Maehara. *Eine Darstellung der intuitionistischen Logik in der klassischen.* Nagoya Mathematical Journal, pages 45–64, 1954. 3, 4, 105, 110

[McAllester 2002] David A. McAllester. *On the complexity analysis of static analyses.* J. ACM, vol. 49, no. 4, pages 512–537, 2002. 138

[McAllester 2003] D. McAllester. *A Logical Algorithm for ML Type Inference.* In R. Nieuwenhuis, editeur, Rewriting Techniques and Applications, 14th International Conference, RTA-03, volume 2706 of *LNCS*, pages 436–451, Valencia, Spain, 2003. Springer. 138

[McDowell 2000] Raymond McDowell and Dale Miller. *Cut-elimination for a logic with definitions and induction.* Theoretical Computer Science, vol. 232, pages 91–119, 2000. 42

[McDowell 2003] Raymond McDowell, Dale Miller and Catuscia Palamidessi. *Encoding transition systems in sequent calculus.* Theoretical Computer Science, vol. 294, no. 3, pages 411–437, 2003. 44, 53

[Miller 1987] Dale Miller, Gopalan Nadathur and Andre Scedrov. *Hereditary Harrop Formulas and Uniform Proof Systems.* In David Gries, editeur, Symposium on Logic in Computer Science, pages 98–105, Ithaca, NY, June 1987. 21

[Miller 1989] Dale Miller. *A logical analysis of modules in logic programming.* Journal of Logic Programming, vol. 6, no. 1-2, pages 79–108, January 1989. 23

[Miller 1991] Dale Miller, Gopalan Nadathur, Frank Pfenning and Andre Scedrov. *Uniform Proofs as a Foundation for Logic Programming.* Annals of Pure and Applied Logic, vol. 51, pages 125–157, 1991. 19, 37

[Miller 1996] Dale Miller. *Forum: A Multiple-Conclusion Specification Logic.* Theoretical Computer Science, vol. 165, no. 1, pages 201–232, September 1996. 2, 59, 81, 131, 137, 138

[Miller 2002] Dale Miller and Elaine Pimentel. *Using linear logic to reason about sequent systems.* In Uwe Egly and Christian G. Fermüller, editeurs, International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, volume 2381 of *LNCS*, pages 2–23. Springer, 2002. 2, 59, 65, 67, 125

[Miller 2004] Dale Miller and Elaine Pimentel. *Linear logic as a framework for specifying sequent calculus.* In Jan van Eijck, Vincent van Oostrom and Albert Visser, editeurs, Logic Colloquium '99: Proceedings of the Annual European Summer Meeting of the Association for Symbolic Logic, Lecture Notes in Logic, pages 111–135. A K Peters Ltd, 2004. 2, 59, 66

[Miller 2007a] Dale Miller and Vivek Nigam. *Incorporating tables into proofs.* In J. Duparc and T. A. Henzinger, editeurs, CSL 2007: Computer Science Logic, volume 4646 of *LNCS*, pages 466–480. Springer, 2007. 2, 4, 21

[Miller 2007b] Dale Miller and Alexis Saurin. *From proofs to focused proofs: a modular proof of focalization in Linear Logic.* In J. Duparc and T. A. Henzinger, editeurs, CSL 2007: Computer Science Logic, volume 4646 of *LNCS*, pages 405–419. Springer, 2007. 16, 18, 88, 92, 125, 139

[Milner 1978] Robin Milner. *A Theory of Type Polymorphism in Programming.* J. Comput. Syst. Sci., vol. 17, no. 3, pages 348–375, 1978. 4, 55, 141

[Milner 1989] Robin Milner. Communication and concurrency. Prentice-Hall International, 1989. 53

[Minsky 1961] Marvin Minsky. *Recursive unsolvability of Post's problem of 'tag' and other topics in the theory of Turing machines.* Annals of Mathematics, 1961. 103

[Momigliano 2003] Alberto Momigliano and Alwen Tiu. *Induction and Co-induction in Sequent Calculus.* In Mario Coppo Stefano Berardi and Ferruccio Damiani, editeurs, Post-proceedings of TYPES 2003, numéro 3085 de LNCS, pages 293–308, January 2003. 42

[Necula 1997] George C. Necula. *Proof-carrying code.* In Conference Record of the 24th Symposium on Principles of Programming Languages 97, pages 106–119, Paris, France, 1997. ACM Press. 37, 52

[Negri 2001] Sara Negri and Jan Von Plato. Structural proof theory. Cambridge University Press, 2001. 70, 74, 75

[Nigam 2008a] Vivek Nigam. *Using tables to construct non-redundant proofs.* In CiE 2008: Abstracts and extended abstracts of unpublished papers, 2008. To appear. 4, 21

[Nigam 2008b] Vivek Nigam and Dale Miller. *Focusing in linear meta-logic.* In Proceedings of IJCAR: International Joint Conference on Automated Reasoning, volume 5195 of *LNAI*, pages 507–522. Springer, 2008. 2, 4, 59

[Nigam 2008c] Vivek Nigam and Dale Miller. Focusing in linear meta-logic: Extended report. Available from http://hal.inria.fr/inria-00281631, 2008. 70

[Nigam 2009a] Vivek Nigam and Dale Miller. Algorithmic specifications in linear logic with subexponentials. Accepted to PPDP, 2009. 4, 85, 127

[Nigam 2009b] Vivek Nigam and Dale Miller. *A framework for proof systems.* submitted, 2009. 4, 59

[Parigot 1992] Michel Parigot. *Free Deduction: An Analysis of "Computations" in Classical Logic.* In Proceedings of the First Russian Conference on Logic Programming, pages 361–380, London, UK, 1992. Springer-Verlag. 3, 4, 60, 75

[Paulson 1989] Lawrence C. Paulson. *The Foundation of a Generic Theorem Prover.* Journal of Automated Reasoning, vol. 5, pages 363–397, September 1989. 2, 59

[Pfenning 1989] Frank Pfenning. *Elf: A Language for Logic Definition and Verified Metaprogramming.* In Fourth Annual Symposium on Logic in Computer Science, pages 313–321, Monterey, CA, June 1989. 2, 59, 81

[Pfenning 2000] Frank Pfenning. *Structural Cut Elimination I. Intuitionistic and Classical Logic.* Information and Computation, vol. 157, no. 1/2, pages 84–141, March 2000. 81

[Pientka 2005] Brigitte Pientka. *Tabling for higher-order logic programming.* In 20th International Conference on Automated Deduction, Talinn, Estonia, pages 54–69. Springer-Verlag, 2005. 1, 2, 22, 23

[Pimentel 2001] Elaine Gouvêa Pimentel. *Lógica linear e a especificação de sistemas computacionais.* PhD thesis, Universidade Federal de Minas Gerais, Belo Horizonte, M.G., Brasil, December 2001. Written in English. 2, 59, 65

[Pimentel 2005] Elaine Pimentel and Dale Miller. *On the specification of sequent systems.* In LPAR 2005: 12th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, numéro 3835 de LNAI, pages 352–366, 2005. 2, 59, 67

[Polakow 2001] Jeff Polakow. *Ordered Linear Logic and Applications.* PhD thesis, Department of Computer Science, Caregnie Mellon, August 2001. 104

[Prawitz 1965] Dag Prawitz. Natural deduction. Almqvist & Wiksell, Uppsala, 1965. 6, 67

[Ramakrishna 1997] Y. S. Ramakrishna, C. R. Ramakrishnan, I. V. Ramakrishnan, Scott A. Smolka, Terrance Swift and David Scott Warren. *Efficient Model Checking Using Tabled Resolution.* In Proceedings of the 9th International Conference on Computer Aided Verification (CAV97), numéro 1254 de LNCS, pages 143–154, 1997. 1, 2, 22, 23

[Roychoudhury 2000] Abhik Roychoudhury, C. R. Ramakrishnan and I. V. Ramakrishnan. *Justifying proofs using memo tables.* In PPDP, pages 178–189, 2000. 53

[Saurin 2008] Alexis Saurin. *Une étude logique du contrôle.* PhD thesis, École Polytechnique, October 2008. 17, 18

[Schellinx 1991] Harold Schellinx. *Some Syntactical Observations on Linear Logic.* Journal of Logic and Computation, vol. 1, no. 4, pages 537–559, 1991. 44

[Schellinx 1993] Harold Schellinx. *A Linear Approach to Modal Proof Theory.* In Workshop on the Proof Theory of Modal Logic, 1993. 101

[Schroeder-Heister 1984] Peter Schroeder-Heister. *A Natural Extension of Natural Deduction.* Journal of Symbolic Logic, vol. 49, no. 4, pages 1284–1300, 1984. 70

[Schroeder-Heister 1993] Peter Schroeder-Heister. *Definitional Reflection and the Completion.* In R. Dyckhoff, editeur, Proceedings of the 4th International Workshop on Extensions of Logic Programming, pages 333–347. Springer-Verlag LNAI 798, 1993. 41

[Sieg 1998] Wilfried Sieg and John Byrnes. *Normal Natural Deduction Proofs (in classical logic).* Studia Logica, vol. 60, no. 1, pages 67–106, 1998. 67

[Simmons 2008] Robert J. Simmons and Frank Pfenning. *Linear Logical Algorithms.* In ICALP (2), pages 336–347, 2008. 138

[Simmons 2009] Robert J. Simmons and Frank Pfenning. *Linear logical approximations.* In PEPM, pages 9–20, 2009. 104

[Slaney 1989] John Slaney. *Solution to a problem of Ono and Komori.* Journal of Philosophic Logic, vol. 18, pages 103–111, 1989. 24

[Smullyan 1968a] Raymond M. Smullyan. *Analytic Cut.* J. of Symbolic Logic, vol. 33, no. 4, pages 560–564, 1968. 3, 4, 60, 79

[Smullyan 1968b] Raymond M. Smullyan. First-order logic. Springer-Verlag, New York Inc., 1968. 60

[Tiu 2004] Alwen Tiu. *A Logical Framework for Reasoning about Logical Specifications.* PhD thesis, Pennsylvania State University, May 2004. 42

[Tiu 2005] Alwen Tiu, Gopalan Nadathur and Dale Miller. *Mixing Finite Success and Finite Failure in an Automated Prover.* In Proceedings of ESHOL'05: Empirically Successful Automated Reasoning in Higher-Order Logics, pages 79–98, December 2005. 21, 42, 43, 45, 141

[Troelstra 1996] Anne S. Troelstra and Helmut Schwichtenberg. Basic proof theory. Cambridge University Press, 1996. 5, 7, 105, 106, 125

[von Plato 2001] Jan von Plato. *Natural Deduction with General Elimination Rules.* Archive for Mathematical Logic, vol. 40, no. 7, pages 541–567, 2001. 3, 4, 60, 70

[Winikoff 1995] M. Winikoff and J. Harland. *Implementing the Linear Logic Programming Language Lygon.* In Proceedings of the International Logic Programming Symposium, pages 66–80, December 1995. 138

Index